
# NAVAL POSTGRADUATE SCHOOL

**MONTEREY, CALIFORNIA**

# THESIS

**MOBILE SECURITY ENCLAVES**

by

Kevin J. LaFrenier

September 2011

Thesis Co-Advisors: Gurminder Singh
John H. Gibson

**Approved for public release; distribution is unlimited**

THIS PAGE INTENTIONALLY LEFT BLANK

| REPORT DOCUMENTATION PAGE | | | Form Approved OMB No. 0704-0188 |
|---|---|---|---|
| Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503. | | | |
| **1. AGENCY USE ONLY** ***(Leave blank)*** | **2. REPORT DATE** September 2011 | **3. REPORT TYPE AND DATES COVERED** Master's Thesis | |
| **4. TITLE AND SUBTITLE** Mobile Security Enclaves | | | **5. FUNDING NUMBERS** |
| **6. AUTHOR(S)** Kevin J. LaFrenier | | | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Naval Postgraduate School Monterey, CA 93943-5000 | | | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| **9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)** N/A | | | **10. SPONSORING/MONITORING AGENCY REPORT NUMBER** |
| **11. SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number _____N/A_____. | | | |
| **12a. DISTRIBUTION / AVAILABILITY STATEMENT** Approved for public release; distribution unlimited | | | **12b. DISTRIBUTION CODE** **A** |

**13. ABSTRACT (maximum 200 words)**

There are currently no access control methods to permit personnel, such as military members, government agencies, or first-responders, access to restricted resources and applications that are only available when certain conditions are satisfied. Such conditions include user authentication, authorized geographic locations, and connections to specific base transceiver stations or base station controllers. This work defines mobile security enclaves, which are designed to provide this access control, are adaptable and compatible with mobile cellular infrastructures, and can operate without being connected to a dedicated back-end network. The goal of this proposed architecture is to permit users who satisfy specific pre-conditions access to resources and applications to which they otherwise normally would not be granted access. An example where this research is beneficial is during crisis response. Disasters require first responders the need to have immediate access to resources available in a specific location. Another example is agencies requiring mobile communication device use on classified networks or to access classified resources. These mobile security enclaves not only provide strict security by authenticating the user and device location, they also prevent access to networks or resources outside of authorized areas and restrict unauthorized users.

| **14. SUBJECT TERMS** Mobile Communication Devices, Mobile Security, Mobile Enclaves, Security Applications, GSM Security Applications, Mobile Base Station Subsystems, Mobile Access Controls, Mobile Authentication, SIM Authentication | | | **15. NUMBER OF PAGES** 95 |
|---|---|---|---|
| | | | **16. PRICE CODE** |
| **17. SECURITY CLASSIFICATION OF REPORT** Unclassified | **18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified | **19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified | **20. LIMITATION OF ABSTRACT** UU |

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution unlimited**

**MOBILE SECURITY ENCLAVES**

Kevin J. LaFrenier
Captain, United States Marine Corps
B.S., United States Naval Academy, 2002

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

from the

**NAVAL POSTGRADUATE SCHOOL**
**September 2011**

Author: Kevin J. LaFrenier

Approved by: Gurminder Singh
Thesis Co-Advisor

John H. Gibson
Thesis Co-Advisor

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

There are currently no access control methods to permit personnel, such as military members, government agencies, or first-responders, access to restricted resources and applications that are only available when certain conditions are satisfied. Such conditions include user authentication, authorized geographic locations, and connections to specific base transceiver stations or base station controllers. This work defines mobile security enclaves, which are designed to provide this access control, are adaptable and compatible with mobile cellular infrastructures, and can operate without being connected to a dedicated back-end network. The goal of this proposed architecture is to permit users who satisfy specific pre-conditions access to resources and applications to which they otherwise normally would not be granted access. An example where this research is beneficial is during crisis response. Disasters require first responders the need to have immediate access to resources available in a specific location. Another example is agencies requiring mobile communication device use on classified networks or to access classified resources. These mobile security enclaves not only provide strict security by authenticating the user and device location, they also prevent access to networks or resources outside of authorized areas and restrict unauthorized users.

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

# LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| AUC | Authentication Center |
| APDU | Application Protocol Data Unit |
| API | Application Programming Interface |
| BCCH | Broadcast Control Channel |
| BSC | Base Station Controller |
| BSS | Base Station Subsystem |
| BTS | Base Transceiver Station |
| EIR | Equipment Identity Register |
| GPS | Global Positioning System |
| GSM | Global System For Mobile Communications |
| HLR | Home Location Register |
| ICCID | Integrated Circuit Card Identifier |
| IEC | International Electrotechnical Commission |
| IOS | International Organization For Standardization |
| IMEI | International Mobile Equipment Identity |
| IMSI | International Mobile Subscribers Identity |
| ISDN | Integrated Services Digital Network |
| LAC | Location Area Code |
| LAI | Location Area Identity |
| MANET | Mobile Ad-hoc Network |
| MCC | Mobile Country Code |
| MCD | Mobile Communications Device |
| MNC | Mobile Network Code |

| | |
|---|---|
| MSC | Mobile Switching Center |
| MSISDN | Mobile Subscribers ISDN |
| MVPN | Mobile Virtual Private Network |
| NSS | Network Subsystem |
| OS | Operating System |
| PLMN | Public Land Mobile Network |
| PSTN | Public Switched Telephone Network |
| RAND | Random Challenge |
| SDK | Software Development Kit |
| SIM | Subscriber Identity Module |
| SRES | Signed Response |
| TMSI | Temporary Mobile Subscriber Identity |
| VLR | Visitor Location Register |
| VPN | Virtual Private Network |

# ACKNOWLEDGMENTS

Many thanks to both of my advisors, Dr. Gurminder Singh and Mr. John Gibson, for making this rewarding experience possible. Thank you both for continuously keeping me on track and having patience with my many questions. Professor Singh, thank you for taking me in and helping me develop this amazing topic and area of work. Mr. Gibson, thank you for always being available for advice and answering questions right away, considering your very busy schedule. The time spent on this has been an experience I will never forget and I hope to maintain many of the skill sets I have learned throughout this research and development.

I would like to thank Charles Prince, the research associate for the networks track within the department. Charles was always able to fix any problems dealing with getting the equipment set up and running for us, while taking time away from his own work. I would not have been able to run my experiments and testing without your help.

I also wish to thank my classmates for all of the assistance as we tackled a trying curriculum. I would especially like to thank Jim McShea for your unselfish commitment in assisting us in many of the topics we struggled through. I know I am not alone in thanking you for this.

Finally, I owe my lovely wife Amy, and our three children—Iris, Ava, and Jack—so many thanks for hanging in there and always having patience with me. I spent a lot of time away the past few years working hard on this and could not have done it without each and every one of you. Thank you.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

The past decade has brought many natural disasters and times of strife requiring the quick response of emergency personnel, also known as first responders. The first responders have been tasked with many different jobs, such as flood evacuations, aiding tornado victims, and setting up and managing aid stations in earthquake-ravaged countries, to name just a few examples. These examples are endless when considering quick response actions required of law enforcement and the military, fire departments and medical staffs, and many other organizations. The coordination and communication of these great missions are largely complicated and often difficult to manage, yet extremely important.

The current generation of wireless networks and services can support the coordination and communication requirements of such missions. There has been an explosion in the mobile device technologies recently and an even greater advancement of the capabilities of these devices. First responders can benefit from these advances by using them within secure wireless networks to aid in their efforts. Security is a necessity in these environments and although many mechanisms are currently available for the networks themselves, they are not sufficient enough to secure an application to the native device.

## A. SECURE ENCLAVES

Valuable services and resources must be available to first responders on-scene. These are often just applications, and must be secure enough to not only share sensitive data without compromise but also prevent unauthorized access. Wireless networks currently do an acceptable job with providing security, but there also needs to be a secure form of access control between the user's mobile device and applications or data.

A mobile security enclave is a method of access control between a collection of valuable assets and the mobile device. The enclave contains these assets and only allows access to the user when certain conditions exist. The conditions must be specific to the

user and device, such as connection to particular cell tower or present within a geographical area. These conditions relate directly to users who often operate with mobile ad-hoc networks. These types of networks should also provide additional security and present solutions when the permanent infrastructure has been damaged or inaccessible. There are currently no mobile security enclaves in use.

## B. OBJECTIVES

Our goal is to develop an infrastructure for mobile security enclaves. The infrastructure will consist of an application to test appropriate conditions under which access should or should not be granted, a method of authentication between the application and mobile device, and the framework required to allow the access from device to sensitive assets (the enclave). The following components of the framework will be presented:

- The testing application is the first instance of allowing secure access to the enclave. It tests whether or not the mobile device is currently connected to a specific base station. If the test is positive, then the authentication between the security enclave and mobile device may begin. Further work will be conducted with other conditions, such as proper GPS coordinates of the device.
- A method of authentication between the enclave and mobile device is proposed. Various schemes are considered, such as a shared key mechanism, to allow for mutual authentication.
- Once the device and enclave are authenticated, assets within the enclave are available to the user. A method of building the enclave is also proposed to include the Android coding required.

## C. ORGANIZATION

Chapter I provides a brief introduction to the proposed work surrounding security enclaves. The introduction includes a short description of what enclaves are and what they can be used for. The chapter also outlines the objectives that will be achieved.

Chapter II provides the reader a description of technologies and concepts used and discussed in this work. The topics range from very broad concepts to precise definitions. This chapter gives the reader a general understanding of the areas of work and focuses on the relevant aspects.

Chapter III examines the necessary framework required to meet the objectives. It covers requirements for the application and concepts, and explains what each component must accomplish. This chapter will also discuss a generalized architecture for providing location-verification of the device, how to incorporate the authentication, and how to design the enclave.

Chapter IV explains the architecture from Chapter III in detail as to how it works and how it is to be implemented. There are three main sections to this chapter, to include the location-verification application, authentication mechanism, and security enclave. The location-verification application is implemented and the code is included in the Appendices. The first section explains the code used and how it relates to the remaining work. The next two sections provide a detailed description of how the authentication mechanism and enclave needs to be developed.

Chapter V is the summary of the thesis with conclusions drawn from the testing as well as proposed frameworks. The chapter also outlines future follow-up work necessary to put these concepts into a working solution and other components that can be added or altered to provide further functionality.

THIS PAGE INTENTIONALLY LEFT BLANK

# II. BACKGROUND

## A. INTRODUCTION

This chapter prepares the reader for follow-up discussions presented in subsequent chapters. It provides a general background and description of cellular communication networks and an analysis of the existing architectures of the relationships between applications, mobile devices, and networks. The discussion begins with basic cellular network operation and components, continues with GSM and related functions, and concludes with information about the proposed frameworks for mobile security enclaves defined in Chapters III and IV.

## B. CELLULAR NETWORKS AND GSM

The need for a wireless infrastructure to support communications and data interactions is now more important than ever with the increased use of mobile devices. Although the massive growth in mobile devices has only occurred recently, the concept and technology has been around for many decades. There is much history behind the evolution of the cellular networks including the current 3G networks (quickly upgrading to 4G); however, this chapter only discusses the emergence of the Global System for Mobile Communications (GSM), since it is the only network used as a test-bed by this thesis.

The basic components of a cellular network are similar regardless of which device, type of network, or service provider is being used. The back-end of any network is the Public Switched Telephone Network (PSTN), which extends out to the Network Subsystem (NSS) that contains the Mobile Switching Centers (MSC), among many other components. MSCs connect to each other and Base Station Subsystems (BSS) that each contain a Base Station Controller (BSC) attached to at least one Base Transceiver Station (BTS), or cell tower. The towers are the components that reach out and "talk" to the mobile devices through the over-the-air interface. These components are discussed more in depth for GSM networks only (Figure 1), following a brief background of GSM.

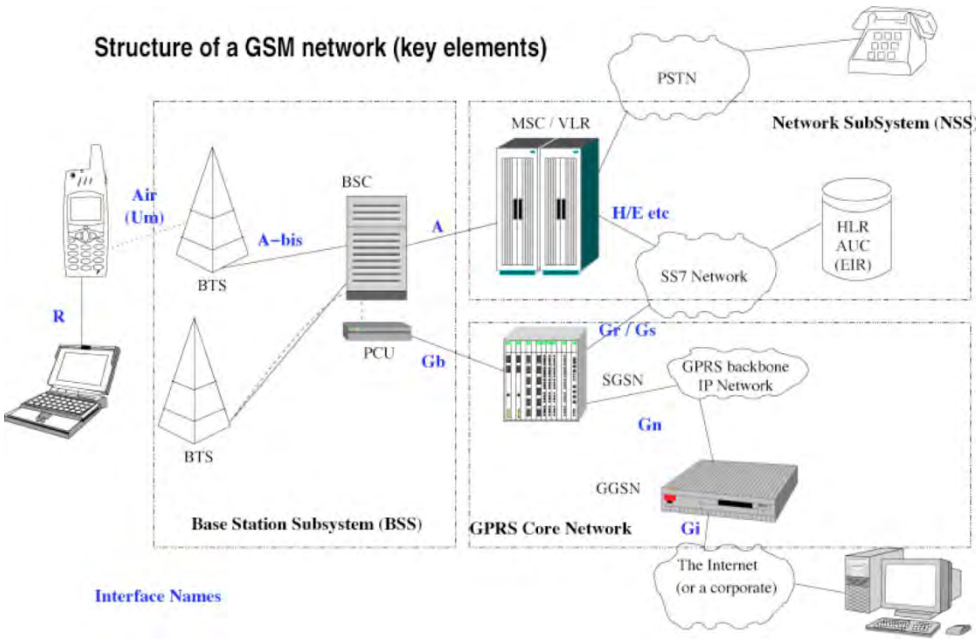


Figure 1. GSM network architecture (From [1])

GSM started out as The Groupe Special Mobile in 1982 by the European Conference of Postal and Telecommunications Administrations (CEPT). The purpose of this group was to develop a standard for a mobile telephone system that could be used across Europe. Five years later, 13 other countries in Europe committed to deploying GSM [2]. The first GSM network infrastructure was implemented in 1991, as well as the first GSM-based call being made. The number of Global GSM subscribers in 1995 exceeded 10 million, 100 million in 1998 [3] and projected to be more than 4.5 billion in late 2012 [4]. The global GSM market share as of 2010 is 83.5% [5].

The use of mobile telephony devices in a network such as GSM is made possible by the PSTN. This is the core infrastructure used worldwide to connect all devices and networks together. It is a massive interconnection of circuit-switched telephone networks. The cellular network is one such telephone network. An example of how the PSTN connects a call from one device to another is listed below.

The GSM Network Subsystem (also known as the Network Switching Subsystem or NSS, or GSM Core Network) connects the PSTN to the BSS and contains the following elements: Home Location Register (HLR), Visitor Location Register (VLR),

Authentication Center (AuC), Equipment Identity Register (EIR) and the MSC. The main roles of the NSS are acting as a switching center and mobility management. The HLR is a database that contains a subscriber's basic information, such as identification and home network. Similar to the HLR is the VLR that contains the current location information of the user to be used in tracking the device and roaming. The MSC is the node responsible for end-to-end routing of voice calls, text messages, and other data services. It is connected directly to the other components, as well as the BSS and PSTN. The AuC authenticates a subscriber's Subscriber Identity Module (SIM) when the subscriber's handset attempts to connect to the NSS. The SIM is explained in more detail below. Lastly, the purpose of the EIR is to monitor and track stolen mobile devices by their International Mobile Equipment Identity (IMEI). An IMEI number is submitted to the EIR, which maintains a list of stolen mobile device IMEI that are monitored for presence on GSM networks.

Base Station Subsystems handle traffic between the mobile devices (also referred to as mobile communication devices, or MCDs) and the NSSs. Each BSS contains a BSC and one or more BTS's. In most networks the BSC will be surrounded by multiple BTS's creating a pattern of cellular coverage, a "cell." The BSC extracts signal measurements used in BTS handoffs and controls radio channels. The BTS simply contains the equipment used for transmitting and receiving radio signals.

GSM networks contain two separate channels by which mobile device users connect. These are an administration (control) channel and data transfer channel, which itself is comprised of various sub-channels. GSM networks are allotted a frequency range, which is broken down into these channels, eight in all. To avoid frequency overlap, the BTSs will divide their area of coverage up into sections. This also aids in mobile device handoff as it moves from cell to cell. BTSs in GSM networks continuously transmit RF signals on a control frequency for MCDs to detect while mobile stations are continuously scanning the forward control channel (FCC) for paging signals from base stations.

The MCD, BSS and MSC are the components that primarily handle phone calls and text messaging. When a MSC receives a request for a connection to a mobile station

in its area, it sends a broadcast message to all base stations under its control. The message contains the number of the mobile station that is being called. The base stations then broadcast the message on all forward control channels. The correct mobile station acknowledges the page by identifying itself over the reverse control channel (RCC). The MSC receives the acknowledgement via the base station and instructs the base station and mobile station to switch to an unused voice channel. A control message is then transmitted over the forward voice channel, which instructs the mobile phone to ring [6]. The mobile device measures signal strength of all detected signals and relays this information to the MSC which will calculates whether or not to hand the MCD over to another BTS or BSC.

The method is very similar when the mobile device initiates the call instead of from the other side of the MSC. The MCD sends a request to its BTS, which is relayed through to the BSC. The BSC routes the request to the MSC, which decides on whether or not it needs to forward the request to a connected BSS, another MSC, or to the PSTN.

The inherent design of cellular networks leads to security problems since tower to mobile device communications are transmitted in the wireless medium, where eavesdroppers can intercept the signals. GSM offers security by encrypting this link with special ciphers. The ciphers used for encryption are integrated into the MCD as a dedicated piece of silicon. The cipher key is derived by the SIM during the authentication process. To authenticate a SIM, the network has to know its identity. As this has to be sent over the air interface, temporary identities are used to counteract the threat of tracing the user's whereabouts [7].

It is important to note here the potential weaknesses in the over-the-air interface of data transfer between the MCD and BTS. This is one valid concern leading to the development of a mobile security enclave for first responders. This interface provides some security, but our proposed enclave would provide a more sufficient means of securing access and data transfer.

### 1. Core Network Communication

Configuring the core network at the MSC can be a difficult task depending on what equipment the first responders are using. This cannot be done on a permanent GSM network without the system's approval, but other means are available for testing purposes as well as implementation. There exist mobile base stations that can perform many of the same functions as the BSC and MSC, and some allow back-end connection to the permanent infrastructures. First responders can use these mobile base stations to set up their own networks on the fly, similar to mobile ad-hoc networks (MANETs), but are different in protocol. This thesis references mobile networks but they are not to be confused with MANETs. Mobile networks allow users to communicate and transfer data via GSM devices as though they are present within a permanent network. The benefit to the mobile BSC/MSC combination is that they can easily be configured and sufficient security methods can be applied. There also exists open source software that allows for the configuration of components of a GSM network if the equipment is not capable.

One such example of open source software is OpenBSC. The OpenBSC project is a software program that benefits researchers in three ways [8]:

- provides for a low-cost test-bed for experimentation and security research with GSM
- the project documents and publicizes any security related issues that are found
- researchers learn more about GSM networks on a lower level, particularly the practical aspects with real-world equipment

The only requirements for OpenBSC are the software program, a GSM BTS and a Linux kernel with mISDN support. The software program must be written in C99 portable code. C99 is a standardization of the previous revisions and extensions of the C programming language leading up to 1999 [9].

There are different modes within this program that implement and act as the BSC, MSC and HLR. With the required equipment, one is able to setup up a configurable GSM network for testing and researching purposes. The project also provides functionality for use of AuCs, VLRs and EIRs. The HLR is a simple instance of an

SQLite database that stores entries of the subscribers to the network. OpenBTS is a similar program to that of OpenBSC, but replaces the NSS from the BTS up. The purpose is to provide a software based GSM access point for GSM compatible devices. In addition to software, there are whole components that can be used in manually configuring GSM network components. These devices are discussed further in Chapter IV.

Some of the focus in this thesis centers on interaction between the MS and BSS. There exists a possibility to manage this interaction from the MSC. The MSC's purpose is linking groups of BSSs and to control call signaling and processing, among other things. The important aspect is the data transfer through the BSS, intended for components such as the HLR, VLR and EIR, which is necessary for location management.

One such data transfer occurs in mobility management. BTSs are periodically broadcasting cell identities to their areas of coverage. Any MCDs within that area of coverage receive that information and relay it to the VLR that is attached to the local MSC, where it calculates signal strengths for other uses, such as handoff. This new location data is sent from the VLR to the mobile device's own HLR via the MSC to update it. The HLR will then send data to the old VLR instructing it to delete the old location info (typically a different MSC) as well as sending the user's service profile to the new VLR, again through the MSC of focus. The data stored in the VLR consists of the International Mobile Subscriber's Identity (IMSI), authentication data, Mobile Subscriber Integrated Services Digital Network (ISDN/MSIDSN) number, GSM services that the subscriber is allowed access to and the HLR address of the subscriber.

### 2. Base Station Subsystem

The BSS's overall responsibilities include: handling traffic and signaling between the MCD and MSC, encoding of speech channels, allocation of radio channels to MCDs, paging, and transmission and reception of signals. These tasks can be handed out,

respectively, to the BSC and BTS combination. As with the MSC, it is important to focus on the data transfer between the MCD and BSS to determine possible authentication or access control schemes.

There is a significant amount of data that travels through the BSC to other BTSs and the MSC. The focus here is how to manipulate that information to develop an appropriate scheme. Other traffic, such as telephone calls and SMS traffic, is ignored for this work. Rather, information that is inherent to the BSC or MCD is the focus and is primarily identification and location related. One important example of this is BSC identification. This identification, otherwise known as the Cell ID, is continuously being broadcast on the broadcast control channel (BCCH) of each BTS [10]. The Cell ID, received by each MCD that is within range, corresponds to that BTS/BSC pair and can be used for updating location and other functions. There are other types of identification being broadcast, as well, such as the Location Area Identity (LAI), neighboring cell information, beacon frequencies, and minimum received signal strengths.

The LAI involves location updating and identification of the MCD. This is a combination of the Mobile Country Code (MCC), the Mobile Network Code (MNC), and the Location Area Code (LAC). Each of these numbers represents a specific location in the network. The first time the MCD is powered up it compares the stored LAI in the SIM to the LAI being broadcast by the BTS. If they are different, the MCD will update its location through the BSC to the VLR, and further up the network to the MSC if the MCD is being served by a different VLR [11]. The LAI is part of the International Mobile Subscriber Identity (IMSI)/Temporary Mobile Subscriber Identity (TMSI).

Another responsibility of the BSS in GSM networks is that of security. There is the potential for compromise of data traffic between the MCD and BTS because of the over-the-air interface. An eavesdropper with the know-how can intercept a call and listen in on the conversation – among many different types of attacks. There are two approaches to addressing this problem: 1) authentication between the MCD and MSC and 2) data encryption between the MCD and BSS [12]. Security within GSM is discussed in the next section, but it is important to know that the BSS does play a minor role in this area.

## 3. GSM Security

The weakest link in GSM security is the over-the-air interface between the MCD and the BTS. The security model for GSM was created to battle this deficiency by offering a method that grants the network and users the ability to avoid sending sensitive identification information over this interface, such as the International Mobile Subscriber Identity, or IMSI [13]. There are two primary methods of securing the air interface – authentication and traffic encryption.

The first step in the security mechanism is to authenticate the user. The purpose behind authentication is not only for security reasons, but it also plays a role in identifying and locating a user in the network. The moment a MCD attempts to connect to the network, data is being transmitted back and forth in attempts to identify the device, the user and their network permissions. This relies on proper identification, which is only possible with proper authentication. Part of the architectural design in this thesis focuses on the authentication for the purpose of properly identifying a user.

An integral component in GSM for providing authentication is the Subscriber Identity Module (SIM). The SIM is an integrated circuit chip fixed to a card (standard terminology, thus—"SIM card") and contains subscriber information, cipher keys and algorithms used in encryption and authentication. The information and secret keys are embedded into the chip during the personalization process with the service provider, and the same information and keys are distributed to the HLR/AuC. The subscriber information contained within it is the IMSI, which is rarely used for security reasons; the TMSI is mostly used in its place. The keys are shared cipher keys between it and the HLR; they are used for authentication with the MSC. The A3 and A8 algorithm contained within the SIM are used for authenticating the MCD to the MSC and for specific sessions, respectively. The A5 algorithm is used for encryption.

Authentication is possible through the use of shared keys between the MCD and HLR. The shared key, $K_i$, is a 128-bit key used to generate a 32-bit signed response, called SRES, to a random challenge, called RAND, and a 64-bit session key, $K_c$. This is all done by the MSC using the A3 and A8 algorithm, respectively [14]. The process

starts with the MCD attempting to connect to the network. The associated HLR will create a set of five triples, each containing a RAND, SRES to that particular RAND based on the $K_i$ and a $K_c$ based on the same $K_i$. It will send these triples to the MSC to be used in authentication with the MCD. The MSC sends the RAND of the first triple to the MCD. The MCD will calculate the SRES with the RAND it just received from the MSC and its $K_i$ and send it back to the MSC. The MSC then compares the SRES it just received from the MCD to the one it received from the HLR. If they are the same, the MCD is authenticated.

Encryption occurs in a similar fashion. Once the MCD is granted access after successful authentication with the MSC, it will create a session key, $K_c$. This key is created with the A8 algorithm using the same RAND challenge it received from the MSC and the $K_i$ stored in the SIM [15]. The BTS has the same $K_c$ from the group of triples it received from the MSC. Encryption over the air between the MCD and BTS occurs on each frame with a different keystream between the two. This keystream is generated using the A5 algorithm initialized by the $K_c$ and the number of the frame to be encrypted.

## 4. MCD Applications

Applications are computer software programs developed for the end-user. These programs come in a wide variety of functions, such as games, finance assistants, and word processors. These software programs can be built with as little as a few dozen lines of code or are as large as millions of lines. There are many different programming languages with which to write applications, such as Java, C++ and Python; programmers need have little to no experience in writing code to develop a simple application (“app” for short).

Applications are not limited to personal computers. They have appeared in personal data assistants (PDAs), cell phones, portable game consoles, and smartphones. Essentially anything with the ability to store and execute code can run an application. Smartphone applications have taken over the digital world with the advent of their “pocket-sized” computers. As of 2011, application downloads for the more popular

devices (Google & Apple) have exceeded the ten billion mark [16]. Some platforms classes are even seeing millions of downloads in a single day.

## C. ANDROID'S APPLICATION PROGRAMMING INTERFACES (API)

Android is a software stack for mobile devices used to execute applications. It contains an operating system (OS), middleware and key applications. Furthermore, there exists a software development kit (SDK) that contains the necessary libraries and application programming interfaces (APIs) for writing applications for Android devices. Figure 2 displays the overall Android architecture and the specific components are described on the Android Developer's website [17].


Figure 2. Android architecture (From [17])

An Application Programming Interface (API) is a set of programming instructions that allow software programs to communicate with one another. The API for a system or program represents a method for others to access the information its system or program

contains. For example, if one was looking to develop a website that pulls information from other sites such as weather, news and sports updates, to display it on one's own site, they would need the APIs for those specific sources. The APIs give the developer the tools necessary to access the information and use it. An example related to mobile devices is an application that displays the latitude and longitude of the device's current position. The location-based application will use the device's APIs to retrieve the coordinates for display.

The Android SDK contains the Android OS APIs. This collection of APIs allows developers to create applications for Android OS devices. Many of the APIs are open to the public and give access to services such as basic telephony data, location specific information (which is used in this thesis), and GSM and CDMA functions. Others are closed to the public and need special permissions for use, or are strictly controlled by the original equipment manufacturers (OEMs).

## D. LOCATION-BASED APPLICATIONS

The global positioning system (GPS) is being put to use in many different facets of technology. It is embedded within automobiles and dashboard-mounted navigation units. GPS receivers have been incorporated in wristwatches for runners and bikers. GPS also made an appearance in the early years of the cell phones. Currently, many smartphone applications depend upon pulling a MCD's GPS coordinates for use in gaming, shopping, social networking, geo-locating, and so on.

Numerous applications (also known as Location Based Services—LBSs) for mobile devices rely on or utilize the location of the device, which can be found through a few different methods. As mentioned above, tracking a MCD is relatively easy to do from the network by use of the MSC and VLR. Both of these components contain location information about the device, specifically the BSS that the device is connected to. Knowing this information, the device can be found by knowing in which cell the device is located. This narrows down the search and the precision is limited only by the size of the cell. Other methods include using localization or triangulation of BTSs and signal strengths. A third method is by using a GPS receiver, which is embedded in the

phone. The LBSs are implemented by applications written for the device. Another example of an application that queries for location information is one that checks for the Cell ID. Every BTS has an area of coverage, called a cell, and an identity. The area of coverage is the extent of the geographic range that a MCD can make a reliable connection to the BTS within. The Cell ID is a unique number assigned to each BTS or sector (cell) of that BTS. This data is used when the MCD periodically sends signal strength and connection quality data to the MSC where it is calculated for handoff determination [18]. Similar to the location based Java classes mentioned above, there are classes that allow a developer to utilize the Cell ID of a BTS that the device is connected too.

## E. SUMMARY

This chapter introduces a high-level discussion on many broad topics concerning mobile devices, the GSM network, and related architectures. The aim is to provide the reader a basic understanding of components and concepts used in this thesis. The general focus is on developing a context, or security enclave, that provides a secure method of connection and contains vital applications within. The following chapters will go further into detail on how these topics are used.

# III. ARCHITECTURE AND DESIGN

## A. INTRODUCTION

This chapter presents a high level description of an architecture that incorporates a location-verification test application, authentication mechanism and the secure enclave. It also offers other possible combinations of components as well as discusses functions and interactions between the concepts, such as security and authentication. The goal is to develop a working solution based off of the infrastructure in this chapter. This development consists of a test application that allows authentication to occur between the application and mobile communications devices (MCDs), which further allows the enclave to be accessed. This architecture is summarized in Figure 3.

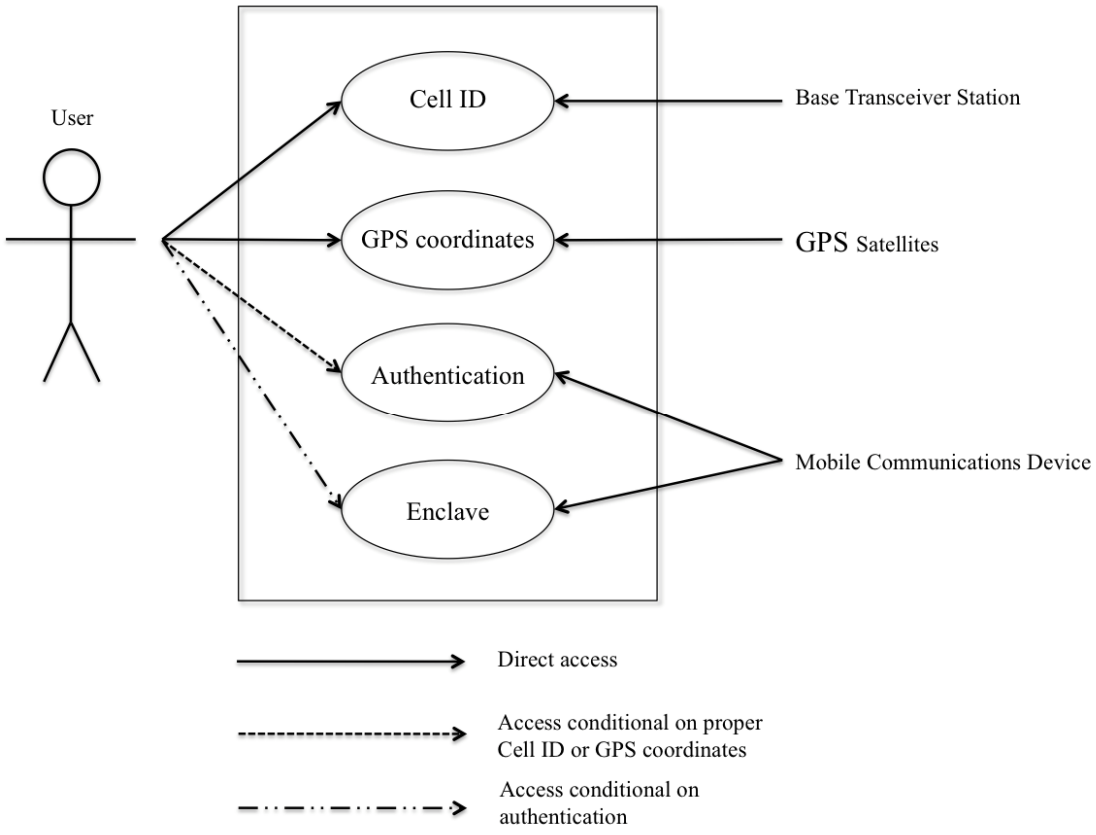


Figure 3. Proposed use case

## B. LOCATION-VERIFICATION APPLICATION

The purpose of this initial test is to verify the user is in a particular location that is authorized to have access to an enclave. This test is the first part of the connection between the user and secure or restricted-access resources. This section discusses two methods of verifying that the user is at a particular location, which will in-turn authorize them access. These locations can be geographic or conditional on connections to base stations since each BTS has a limited range for its transmissions.

The proposed test application is designed for the Android operating system and GSM network. The Android software development toolkit (SDK) allows for applications to be built by using its application programming interfaces (APIs) and the Java programming language [19]. The Java language is object-orientated, making use of objects and classes, which are used in development of this test application. For an in-depth explanation of the Java programming language, visit Java's on-line programming tutorials (http://download.oracle.com/javase/tutorial/index.html). Additionally, specifics on the code, phone and base station used in testing is presented in Chapter IV. The important point here is that the Android OS contains classes within its SDK that allows for an application to obtain the Cell ID of a base station that the MCD is connected to. The Android SDK also allows for access to GPS coordinates.

The application's operation is straightforward for the user. The user opens the application on the MCD and executes a "Connect" function, which is a simple command that instructs the application to perform its test. The application pulls the necessary information from either the base station or the GPS receiver in the MCD to determine authorization. This authorization is initially written into the application during the programming phase, and can be only changed statically. If the test passes, then the application will advance to the next portion in the architecture – authentication of the application to the MCD in order to connect to the enclave in a secure session. Otherwise, if the test fails, then the application will just notify the user that the conditions have not been met.

The following discusses the two methods of testing in the application:

### 1. Cell ID

In this test, the application is checking to see if the MCD is connected to the appropriate base station. As explained in Chapter II, each base station is assigned a unique identification number by the telecommunication services provider, which distinguishes it from others. The identification is broadcast on the broadcast control channel (BCCH) of each BTS, and the MCD receives this data from the BSC to which it is connected, so the application simply reads that ID from the MCD. The application contains a function that directs further action if the Cell ID read is one of the "authorized IDs". These authorized IDs will be coded into a list within the application for cross-referencing. An example of the pseudocode for this test reads:

```
get CELLID;
if CELLID == 12345
{
      do AUTHENTICATION;
}
else
{
      exit;
}
```

It is also possible to change the configurations of some mobile base stations from the network management level, including the Cell IDs. Thus, this Cell ID check is not limited to just permanent GSM infrastructures.

### 2. GPS Coordinates

This test application obtains the GPS coordinates from the receiver in the device. A similar method is written to get the coordinates but instead of cross-referencing an authorization list, the application will perform a calculation. An example of this would entail limiting access to a few city blocks. The coordinates that outline the collection of authorized city blocks are written into the application. When called upon to test a location, the application performs a simple calculation to determine from the users

current GPS coordinates, if they are within the authorized blocks. The following pseudocode gives a simple example of what the calculation would be:

```
get LATITUDE, LONGITUDE;
if LATITUDE < 35.800 AND > 35.700
{
      if LONGITUDE < 121.800 AND > 121.700
      {
            do AUTHENTICATION;
      }
}
else
{
      exit;
}
```

This method has the advantage in that GPS coordinates cannot be spoofed, unlike Cell IDs, but has a drawback of potentially being in a location that does not receive GPS signals. This method is implemented statically and would be useful in situations where the authorized locations are known in advance and do not change.

## C. AUTHENTICATION OF APPLICATION TO MCD

The purpose of the overall framework is to provide a secure method of granting access of vital resources and applications, contained within an enclave, to first responders and emergency personnel. The section above discusses an architecture needed to provide the first form of security by checking location of the device. This section discusses the second form of security – authentication of an application and the MCD.

Both methods of location verification and authentication occur in the same application. Once the application verifies that the MCD is in the proper location, either by GPS coordinates or connection to a specific BSS, then it must authenticate with the MCD. This authentication provides another layer of security for access to the enclave, ensuring that the individual and device attempting to access the enclave is authorized. The authorized individuals and devices are pre-programmed into the application, can be pre-loaded onto a removal storage device or via another means such as accessing the network. This architecture only focuses on the application containing the authorizations.

The primary authentication mechanism is challenge-and-response. The application must challenge the user and dependent on the user response, grants access to the enclave. The first component in this process must be a valid piece of identification. There are a few different forms of identification available to test against on a GSM-connected MCD, to include the IMEI, ICCID, IMSI, and shared keys on the SIM cards, to name a few. Each of these IDs can be retrieved with the proper coding in Android, but the most secure is to use the shared keys contained within the SIM card.

In order to use this method of authentication, the application authenticates the user by sending a challenge to the SIM invoking a response. The challenge is a 128-bit random number (RAND) and is generated by the application. The SIM contains algorithms that combine a shared key, $K_i$, and the RAND to form a response and session key, $K_c$. The application contains similar algorithms and with a copy of the same shared key, generates its own response and $K_c$. Additionally, shared key and IMSI (or some other form of identification) matched pairings need to be stored within the device so that the application will know which shared key to use for the present SIM. Authentication is successful when both responses match. Once authenticated, the application will send a request to open the security enclave. This request contains the challenge, the IMSI and the generated $K_c$. This is the only way to open up the enclave.

## D. ENCLAVE STRUCTURING

The third component to the architecture is the security enclave. The important aspects of this element are strict security features that must be in place for it to be accessed and the fact that it contains vital resources for the users, such as applications or data. The enclave is application-based as well, which offers a few different security methods that prevent unauthorized access. Another benefit of being application-based is that updates can be sent to the devices that contain the enclave similar to that of other applications on the network.

The primary aspect of the enclave is the means of opening it. There are alternate means of structuring the enclave such as building it into a separate partition of the hard drive, developing a separation kernel for it or embedding it within an application. This

work focuses on the application method. The application will be written such that the only way it can be started is by a request from the authentication application above. Sending an intent, or coded message, from the authentication application to the enclave to open it, does this. The intent contains user information, a session key and a specific request to start the enclave application. The enclave will first run the A3/A8 algorithm to produce the $K_c$, similar to that generated during authentication, as described above. It will compare both session keys, and if they are equivalent, will grant the enclave access for the user.

Another form of security for this method that is included in the application is the use of developer certificates. Restrictions can be applied within the code that forces the application to verify both applications (location-verification/authentication & enclave) contain the same developer certificate before allowing access. This method prevents other applications from being granted access by sending their own requests for access to the enclave.

Once the enclave is successfully opened, the user will gain access to the contents contained inside. The application is written such that the user will be presented with a list or separate desktop on the device that contains the resources. The coding for this will be similar to that of the intent transfer between the authentication and enclave applications.

## E. SUMMARY

This section presented a high-level aspect of an architecture that allows access to a secure enclave. The three main components of the design were outlined to include the location verification, authentication and the enclave. All three components are application based and provide a high level of security. The next section will go further into details on how the design will be implemented. Figure 4 represents a generalized flow chart of this architecture.

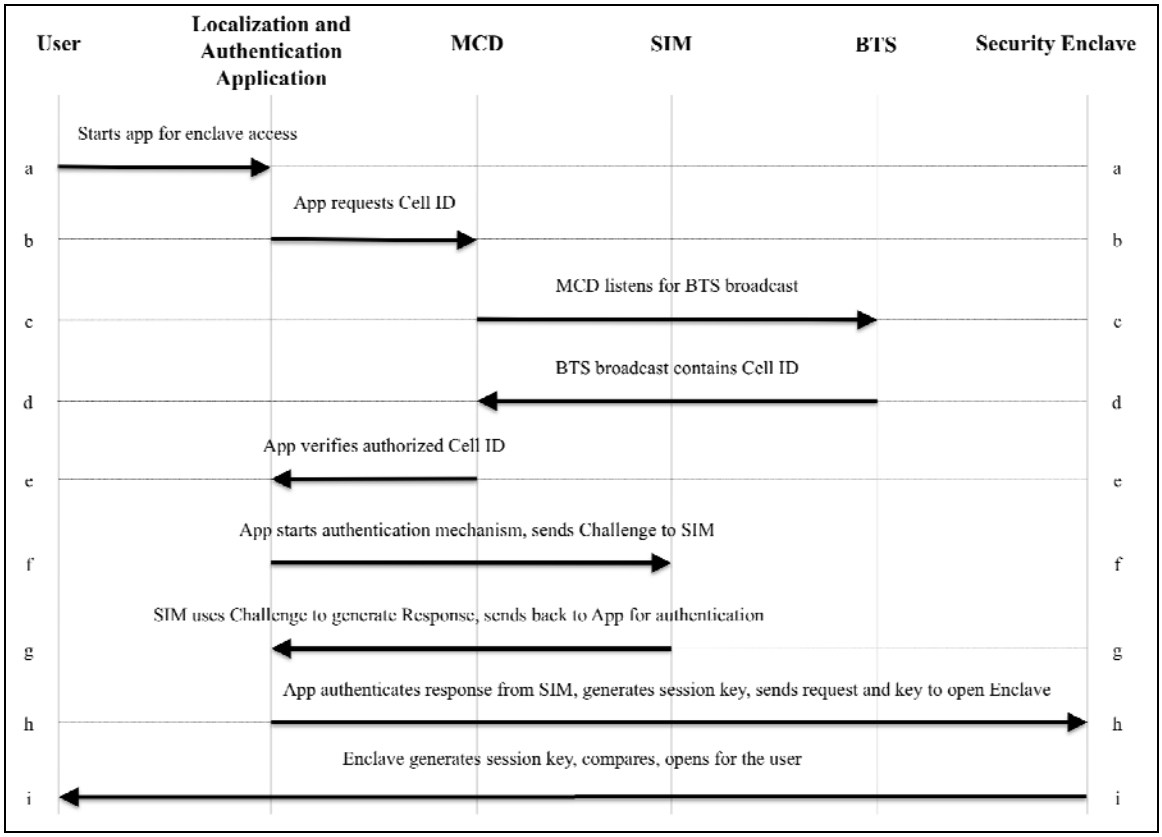


Figure 4. Architectural flow chart

THIS PAGE INTENTIONALLY LEFT BLANK

# IV. IMPLEMENTATION

## A. INTRODUCTION

This chapter describes, in detail, the three main elements of the security enclave architecture. The first section presents relevant code examples with descriptions regarding the written and tested location-verification application. That section explains how the application works and how it performs the verification. The next two sections lay out the required architecture for developing the remaining two aspects of the overall design – authentication to the MCD and the security enclave. Both sections contain the necessary building blocks for implementing the design.

## B. LOCATION-VERIFICATION APPLICATION

The application for verifying location was written for Android devices, firmware version 1.6 and up. The application was installed and tested on an Android Developer Phone (ADP) 2, Google's Ion (HTC Magic). The code was written using Eclipse, which is a Java Integrated Developer Environment (IDE), and it has the Android SDK starter packages and Android Development Tool (ADT) Plugin installed.

LGS's Tactical Base Station Router (TacBSR) Pico was used to represent the GSM network aspect. This architecture is designed to work with any GSM configuration that contains GSM specific elements such as a BTS Cell ID. The TacBSR that is used was designed for Emergency Communications Systems (ECS) applications by providing an alternate GSM network for a pre-provisioned list of subscribers. It was also designed for Search and Rescue (SAR) operations, allowing personnel to communicate to GSM users [20]. It is capable of operating as a BSS, MSC, VLR and HLR, that allows for manual configuration. Its graphic user interface (GUI) allows for administrative functions that would be available in the network as well. Functions specific to this thesis include assigning the Cell ID and verifying IMEI and IMSIs.

This location-verification application contains three main activities (screens): "Main" activity, "GPSActivity" and "CIDActivity." The fourth activity, "AuthActivity," is just a placeholder at this point for future progress and contains minimal functionality. The application is accessed by selecting an icon from the list of applications on the device. The application opens to the first screen (Figure 5). This screen has two options for the user to select: "Get coords" and "Get CellID". Both options represent a means of verifying their present location and will take the user to the respective GPS-coordinates verification or Cell ID verification screen (Figure 6). For testing purposes, these two activities give the option to see what the device's current coordinates and Cell ID are. There is an "Authenticate" button on both of these verification activities that will take the user to the authentication activity. This advance can only occur if the user is within a specific geographic region or connected to a specific BTS. If neither condition is satisfied, the user will be unable to advance to the authentication activity. This is the main function of the location-verification application; without authorized coordinates or Cell ID, the user is unable to authenticate, which is required to open the enclave.

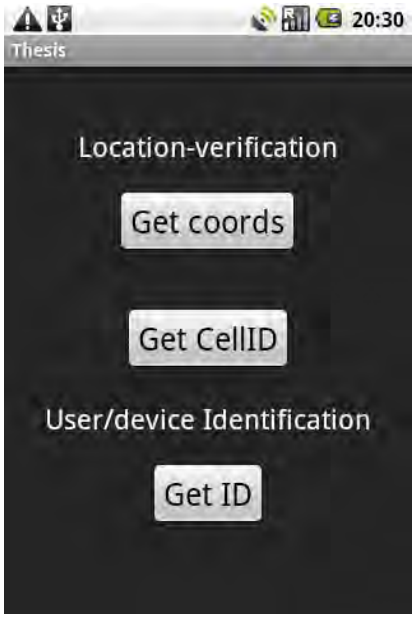


Figure 5. Location-verification application home activity

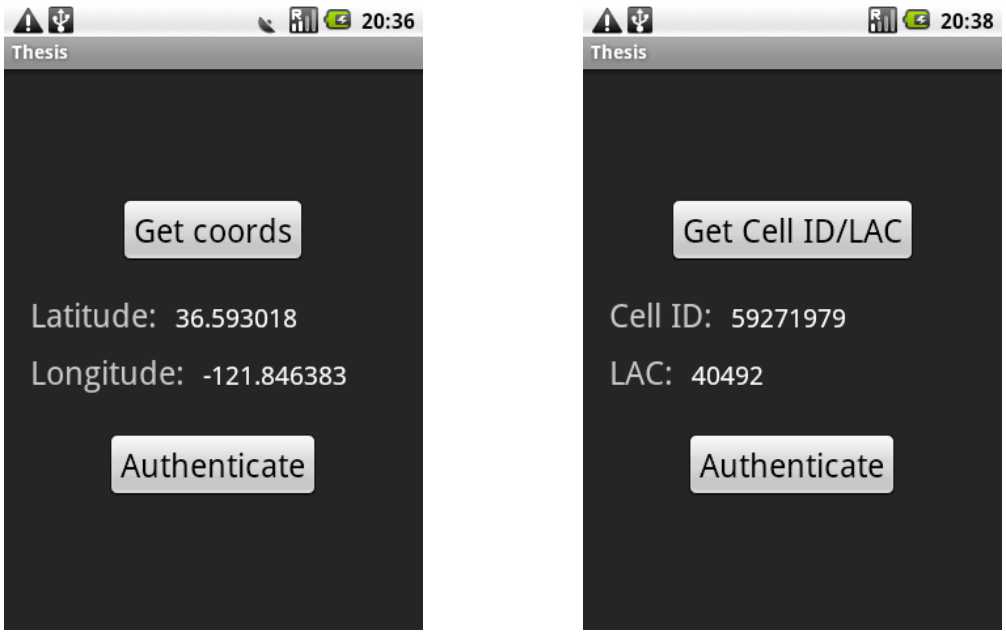


Figure 6. Location-verification application GPS and Cell ID check

The user will advance to the authentication activity (Figure 7) when either condition is met. This activity simply contains an "Authenticate" button. This is required to open the enclave; the authentication mechanism will commence once the user depresses the button. A message will be displayed to the user with authentication results. If authentication was a success, this application will send a request to open the security enclave. If it was unsuccessful, a message is displayed to the user saying so. Both the authentication mechanism and enclave are discussed in further sections.

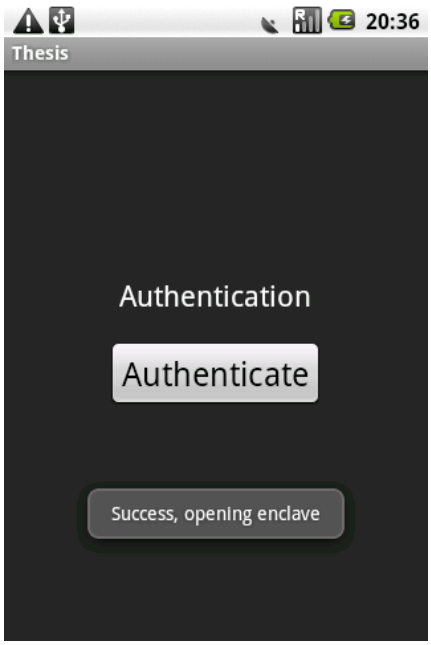


Figure 7. Location-verification application authentication activity

### 1. Main Activity

The Main activity (Appendix A) is the home screen for this application. It contains code to provide functionality to both buttons. The code simply consists of setting an `OnClickListener` to both buttons, which direct the application to the respective activity. For example, the following code outlines the requirements for putting functionality to a button:

This instantiates the buttons:

```
Button gpsButton, cidButton;
```

This sets a listener to the button, which ties the touch-screen actions to the code:

```
gpsButton = (Button) this.findViewById(R.id.gpsBtn);
gpsButton.setOnClickListener(this);
```

And finally, this conditional sets action to the depressed button:

```
Button clickedBtn = (Button) v;
  if (clickedBtn == gpsButton)
    {
    Main.this.startActivity(newIntent(Main.this,GPSActivity.class));
    }
```

The "`GPSActivity.class`" in the previous code represents another activity to open. All buttons in the application are coded in similar fashion. In the example code set above, the two buttons were built into the application such that, when depressed on the screen by the user, the application would advance to that specific activity.

### 2. GPSActivity

The first option from the Main activity is to verify the user's GPS coordinates are authorized. The `GPSActivity` (Appendix B) will open when the `Get coords` button is depressed on the Main activity. The `GPSActivity` is responsible for obtaining the MCD's most recent GPS coordinates and performing a calculation to verify authorized location. This activity is divided into two separate actions: obtaining coordinates and verifying.

Android contains a package `android.location` that has three classes used for obtaining GPS coordinates: `android.location.Location`, `android.location.LocationManager` and `android.location.LocationProvider` [21]. This activity creates an instance of the class `LocationManager`, which provides access to the MCD's location services. Once the activity has access to the location services, it is then able to register for requesting location updates for use in the application. This allows the device to start obtaining position fixes. Once the device has a fix, the activity can request the respective latitude and longitude. This application takes both of these integers and displays them in a `TextView`, as well as using them to perform the calculation used in verifying location authorization. A `TextView` is simply a placeholder for text to be displayed on the screen. The following code outlines the major components for obtaining GPS fixes:

The first step is instantiating the class for requesting device services:

```
LocationManager locMgr;
```

Next will be registering for and requesting location updates. The two values listed in the request for location update determine the minimum time (in milliseconds) and distance (in meters) that the application delays before obtaining a new fix:

```
locMgr = (LocationManager)getSystemService(LOCATION_SERVICE);
locMgr.requestLocationUpdates(LocationManager.GPS_PROVIDER,10000,5.0f,
onLocationChange);
```

At this point, the activity can request the last known location,

```
Location loc = locMgr.getLastKnownLocation(LocationManager.GPS_PROVIDER);
```

and break it up into separate latitude and longitude integers:

```
lat = loc.getLatitude();
lon = loc.getLongitude();
```

Displaying these onto the screen requires outputting them to a `TextView`:

```
TextView myView1 = (TextView) findViewById(R.id.Latitude);
myView1.setText("" + lat);
```

The second part of this activity is verifying the user's location. This requires a set of GPS coordinates as boundaries of an authorized area. These boundaries are written into the code by the developer. The activity performs the calculation when the user clicks the "Authenticate" button on the screen. The activity will obtain the last known latitude and longitude values and compare them against the input boundaries. If the user is within the boundaries, then the application will allow the user to access the authentication activity. If the user is not within the boundaries, the activity will display an error message to the user stating that condition. An example of the calculations is as follows ("`lat`" and "`lon`" are the current latitude and longitude coordinates, the others, such as "`NWlat,`" are manually input authorized coordinates by the developer or organization):

```
if ((lat < NWlat) && (lat < NElat) && (lat > SWlat) && (lat > SElat) &&
      (lon > NWlon) && (lon > SWlon) && (lon < SElon) && (lon < NElon) )
{
      GPSActivity.this.startActivity(new Intent(GPSActivity.this,
AuthActivity.class));
}
else
{
      GPSActivity.this.finish( );
}
```

If the user is not within the authorized boundaries, the activity will shut down and send the user back to the `Main` activity, otherwise they will be advanced to the `GPSActivity`.

In addition to the few examples of code above and Appendix A, there are a few other components that are needed for this activity to work – entries in the "manifest" file

and the xml code. The xml for this activity is straightforward and can be referenced in Appendix C. Both permissions are needed within the `manifest` file to allow the application to obtain network and location information [22]. The following is an example of this portion of the `manifest` file:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COURSE_LOCATION" />
```

The `manifest` file can be referenced in Appendix D.

### 3. CIDActivity

The second option of the `Main` activity will take the user to the `CIDActivity` (Appendix E). Like the `GPSActivity`, the first option is to present the user with the current network information, such as the Cell ID and Location Area Code (LAC). The LAC in this case is for testing purposes only. The Cell ID, if one is available, will be displayed in the proper format; otherwise the application will notify the user there is no connection. Also similar to the `GPSActivity`, there is an "Authenticate" button, which directs the user to the authentication mechanism as long as the MCD is connected to an authorized Cell ID.

Android contains two packages containing classes that allow the programmer to obtain network information, such as the Cell ID. These packages are `android.telephony` and `android.telephony.gsm`. Each contains the required classes of `android.telephony.TelephonyManager` and `android.telephony.gsm.GsmCellLocation`, classes respectively. Similar to the GPS location request in the `GPSActivity`, the activity must request services of the phone to access this network data. This is done so by:

```
TelephonyManager tm = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);
```

This allows access to the following class and an instantiation of it to be used for data retrieval:

```
GsmCellLocation loc = (GsmCellLocation) tm.getCellLocation();
```

Within this class is the method that retrieves the Cell ID from the MCD:

```
cellID = loc.getCid();
```

The remaining portions of this first function in the activity consist of displaying this data to a `TextView` for the user.

The second portion of the activity is whether or not to allow the user to advance to the authentication mechanism. The application should only allow this to occur if the user is connected to an authorized Cell ID. The list of authorized Cell IDs is written into the code or can be stored elsewhere such as the back-end network. These IDs will be cross-referenced when the user depresses the "Authenticate" button to compare the current Cell ID to the list. If there is a match, then the user will be directed to the `AuthActivity`, otherwise the activity will terminate. The following is the conditional loop verifying authorized Cell ID, where in this example the "ID" is a constant representing where the true value would be checked:

```
if (clickedBtn == authBtn)
{
	if (cellID == ID)
	{
	CIDActivity.this.startActivity(new Intent(CIDActivity.this,
AuthActivity.class));
	}
	else
	{
	CIDActivity.this.finish( );
	}
}
```

The only requirement in the manifest file is the same course location access permission that was required by the `GPSActivity` as well. The xml file can be referenced in Appendix F.

### 4. AuthActivity

This activity only contains a button with a notification tied to it. The architecture ties this to the authentication mechanism that is discussed below. The activity and xml for this can be referenced in Appendices G and H, respectively.

## C. AUTHENTICATION OF APPLICATION TO MCD

This section outlines the framework that is needed for the authentication mechanism within the application. It must not be accessible until the above conditions are met, and must allow only those users that contain the authorized identifications. These identifications will come from the SIM and the MCD. This application will authenticate the user by these identifications and establish a session key that will provide another layer of security when attempting to access the security enclave.

This section is comprised of three main parts. The first covers concepts used in the authentication mechanism. The second explains the authentication algorithm used in the application. The third section explains how communication to and from the SIM is possible. The last section will also explain the transition from the authentication portion of this application to the enclave.

### 1. Required Elements And Concepts

The first concept to understand for this mechanism is the process by which authentication occurs between the MCD and HLR/AuC. The general mechanism for this is "challenge and response." In review, the GSM network uses this scheme where the HLR/AuC is creating the challenge and the MCD (SIM, specifically) is providing the response. The authentication mechanism outlined in this proposed architecture closely resembles that of the GSM process, but contains a few different elements.

The components that make up the GSM authentication process are the SIM, MCD, BTS, BSC, VLR, MSC, HLR and AuC. The process starts at subscription between a user and the service provider, when the individual user is associated to a Subscriber Authentication Key, $K_i$, along with an International Mobile Subscriber Identity (IMSI) [23]. This $K_i$ is stored in the SIM and on the network side in the Authentication Center (AuC). In some cases, this AuC is integrated with an HLR. The identity of the user or MCD must be established each time it attempts to connect to the network [24]. The MCD sends its Temporary Mobile Subscriber Identity (TMSI) to the VLR, and possibly further to the HLR, to properly identify itself to the network. Recall

that the TMSI is used instead of the IMSI in most cases as an attempt to protect the user's identity when transmitting over the unsecure air-interface. The VLR will then send an authentication request to the HLR/AuC to initiate the authentication process.

This process of authenticating is based on the A3 algorithm, which occurs in the SIM and at the HLR/AuC. The HLR/AuC will generate a random, non-predictable number, called the RAND, to be used as a challenge. It will combine the RAND and the user's $K_i$ in the A3 algorithm to generate a Signed Response, or SRES. The SRES and RAND are sent back through the network, the RAND to the SIM and SRES to the VLR. The SIM uses the same embedded A3 algorithm to combine the RAND that it received from the HLR/AuC and its $K_i$ to generate its own SRES and a session key, $K_c$, used by the MCD for encryption. The SIM will respond to the challenge with this SRES, and the two SRESs (from the SIM and HLR/AuC) are compared at the VLR. The MCD is only allowed access to the network if they match. Since only the SIM and HLR/AuC contain the user-specific $K_i$, this provides the means of authenticating the user to the network.

### 2. Application-embedded Authentication Algorithm

This mechanism closely follows that used in the GSM network. The process centers on the SIM algorithms; and SRES generation is the same. The difference is in the way the RAND is delivered to the SIM and the fact that the application itself is playing the role of the HLR/AuC. The application will also conduct the SRES comparison between its own generation of the response and the SIM's response, as was done by the VLR originally. The same shared key, $K_i$, is being used. The RAND and SRES are of the same format. From the aspect of the SIM, everything remains the same as in a typical GSM network.

The first step in this authentication scheme is the algorithm within the application. Both the A3 and A8 algorithms, which are used in SIM cards for authentication, are not publicly available, but are still in use [25]. Additionally, GSM Public Land Mobile Network (PLMN) managers have direct access to these algorithms because they are required to operate the myriad of devices on their networks [26]. This means that the

algorithms are still being deployed in SIMs and that there is availability for certain organizations to use or implement versions of A3/A8 algorithms, or other compatible means. These other compatible algorithms are available to provide further security and exist through leaked documents, reverse engineering, and extensive cryptoanalysis on these secret algorithms. One such example is shown in Appendix J. Furthermore, the 3rd Generation Partnership Project (3GPP – www.3GPP.org), a consortium of partners who develop technical specifications and reports, as well as maintenance and development for GSM, have publications specifying the minute details of such compatible algorithms for A3/A8. One of the more frequently used algorithms is the COMP128.

COMP128 is a derived algorithm that is compatible to the A3/A8 algorithms in use in the GSM network [27]. Many versions of this algorithm are available for download across the Internet for implementation. An example of this algorithm, developed by the persons who are responsible for the reverse engineering of it, is posted in Appendix H, coded in C. The functionality of this algorithm requires the same SRES and $K_c$ produced by its algorithm and that of the SIM. Both will produce the same 32-bit SRES and 64-bit $K_c$.

This process starts with the Cell ID or GPS-coordinates successfully completing verification and the user now having access to the "Authenticate" button. Depressing this button triggers the mechanism to start. The authentication activity will query the SIM's IMSI (or MCD's IMEI, or any other form of user identification) to cross-reference a table within the application to find the appropriate $K_i$. This table will be programmed into the application, but may be accessed via other means. Upon a successful query and table match-up, the activity will produce a RAND and start the authentication algorithm (COMP128 or other variant) with it and the $K_i$ to output an SRES. The activity is ready for the next step – polling the SIM for authentication.

### 3. Communication With the SIM

Receiving the RAND and generating the SRES, from the aspect of the SIM, should not change in this application scheme. The SIM receives the command to run the

algorithm, uses its $K_i$ and RAND as input, and returns the SRES. Instead of talking to the network, however, the SIM is communicating indirectly with the application's authentication algorithm within the associated activity. For a clearer understanding of how this works, the discussion will start at the SIM and progress to the application.

The SIM is the component in the GSM network that contains specific user identity and the algorithms required for authenticating them. It contains a microcomputer that consists of a CPU and three types of memory [28]. The masked programmed read only memory (ROM) usually contains the operating system of the card, the code for the GSM application and the security algorithms A3 and A8. The other types of memory, random access memory (RAM) and the electronically erasable programmable ROM (EEPROM), also take part in the authentication process by buffering data for transfer and storing the data, respectively. The OS of the card controls access between the SIM and MCD and network. This access is typically reading or updating the data on the card; reading is the access required for authentication [29].

Special commands are sent from the MCD to the SIM to interact for various functions, such as instructing it to run the authentication algorithm. The commands are contained within Application Protocol Data Units (APDU). APDUs are of two types: command APDUs or response APDUs [30]. The specific format is outlined by the International Organization for Standardization (IOS) and the International Electrotechnical Commission (IEC) Standard IOS/IEC 7688. In general, each APDU contains specific instructions and data for both the SIM and MCD. In the case of authentication, the MCD will transmit a command APDU containing instructions to run the authentication algorithm and the RAND will be contained within the data portion of the unit. The SIM will execute the appropriate algorithm and transmit a response APDU containing the SRES as part of its data unit.

The authentication portion of the application not only creates the RAND and generates the SRES, it also builds the associated APDU to send to the SIM. Proper constructing of the data unit is specifically outlined in the IOS/IEC 7688, in terms of which commands to use to invoke the SIM's algorithm. Sending the APDU to the SIM

requires APIs that are not native to Android. These open-source APIs, specifically the "SmartCard API," are available from the Secure Element Evaluation Kit for the Android platform (seek-for-android) [31].

These APIs are necessary because of the location of the SIM within the architecture of the Android system (Figure 8). It is connected to the baseband processor and the Android system runs on the application processor. In order to communicate with the SIM, the baseband processor must issue AT (attention) commands. These commands are defined by the 3GPP 27.007 specifications. The AT commands, in turn, will transmit the necessary APDUs to the SIM (such as the command APDU and response APDU) [32].

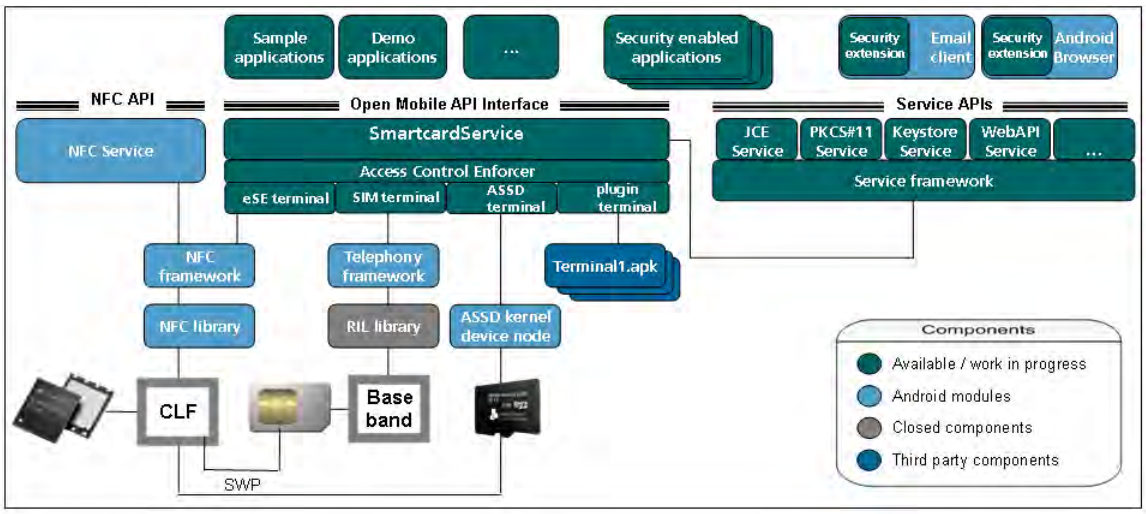


Figure 8. Android system architecture (From [30])

The remaining procedure for authenticating lies in the authentication activity of the application. Upon receiving the response APDU, the activity will use the received data element, which consists of the concatenation of the SRES and $K_c$. Simple parsing can be coded to split these two elements for SRES comparison. With successful authentication, the activity reports authentication and approves access to the enclave.

### 4. Transition To Enclave

This application, which contains both the location-verification and authentication functionalities, will send a request to open the enclave once authentication is successful. The request will contain the enclave's specific identification, and the $K_c$ and RAND used in the algorithm above.

## D. ENCLAVE STRUCTURING

The enclave contains all the necessary resources and applications that are needed by first responders and other personnel for a given response situation. Secure access to these elements is granted by a few different layers, some mentioned above and a few more yet to be discussed. The user that wishes to access these important resources must first be in the authorized location, then authenticated, and a further means of authentication will occur for the enclave to open, but is not the only access control method remaining. There are permissions built into the code that restrict the enclave application from even starting if an unauthorized user is attempting access.

This section will explain the required application framework that makes up the enclave. The first part discusses the communication links between the previous application and this one. The second part goes into detail about the additional securities built within the enclave, followed by the framework that contains the other resources and applications.

### 1. Transition From Authentication To Enclave

Android applications contain four core components that are essential building blocks: activities, services, content providers, and broadcast receivers [33]. All four components offer the system and user (in some cases) an entry into an application. The only component that is of concern for this discussion is the activity.

Activities are simply the individual screens that the user can interact with in the application. They are independent of each other and can be accessed by other applications or activities. This is how the enclave is opened from the location-verification and authentication application. The enclave consists of a main activity and its sole function is to start the enclave application, perform further authentication and verify permissions are appropriate. Activities are started by using the `startActivity(Intent)` function, where the "`Intent`" is a way to describe which specific activity to start.

The system must read the application's manifest file before the application starts. This file contains all the components within the application, such as individual activities. Here the developer must declare the entry point into the application and specify that particular activity be listed in the system's application launcher. This is done by using intent filters; below is an example of how that section of the manifest file will read:

```
<activity android:name=".Main"
                  android:label="@string/app_name">
             <intent-filter>
                 <action android:name="android.intent.action.MAIN" />
                 <category android:name="android.intent.category.LAUNCHER" />
             </intent-filter>
</activity>
```

If these declarations are not in the manifest file, the application will not start when called explicitly by the intent [34].

Intents are messages used to activate the core Android components. They are passive data structures that hold an abstract description of the action to be performed [35]. They also have the ability to carry small amounts of data. The authentication portion of the previously described application utilizes an intent to start the enclave's Main activity. This intent takes the form:

```
Intent intent = new Intent(Intent.ACTION_MAIN);
intent.setComponent(new
       ComponentName("com.example.enclave","com.example.enclave.Main"));
startActivity(intent);
```

The first line initializes the intent and declares the action to take, as starting up an initial activity. The second line sets a component name to the intent so the activity to be performed knows where to find it. The component name has two elements – the Android

package name (the application name in this case) and the name of the class, or activity. The last line details which action needs to take place—start the activity "Main" that resides in the "enclave" application.

### 2. Enclave Securities

At this point, the Main activity of the enclave has opened for the user after successful authentication. The enclave contains two further security functions to ensure that the application was not opened errantly or by unauthorized means. The first method is verifying certain permissions within the manifest file. The second method is executing another authentication algorithm.

The manifest file provides another function for the enclave besides declaring its components to the system upon startup. All Android applications must be signed with a certificate whose private key is held by the developer [36]. The purpose of this is distinguishing application authors. It also provides a level of security by setting permissions within the application to only grant access based on this developer identification.

There are four protection levels in Android: normal, dangerous, signature, and signature or system. All four levels have varying degrees of security and function. By setting the application to a "signature" level, the application will only open if the requesting application contains the same developer certificate. Setting this level requires a manifest entry:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.enclave" >
    <permission android:name="com.example.enclave.Main"
        android:protectionLevel="signature" />
</manifest>
```

The other form of security is authentication, using the authentication portion of the previous application. The intent that is starting the enclave activity will contain the RAND that was previously generated in the SIM authentication and the resulting $K_c$. The same algorithm used in the authentication will be used here. The enclave needs to obtain the SIM identification first to determine which $K_i$ to use, which can be queried in the

same fashion as before. The difference between this instance of authentication and the authentication of the SIM is comparison between the session key that was generated instead of the SRES. Only the application that provided the RAND along with the intent to start the enclave will produce a matching $K_c$. Once both of these conditions are met, the enclave will open full access to the user.

### 3. Enclave Framework

The framework of the enclave can take the form of a list or appear as a separate desktop that contains the resources and applications. The methods of accessing the resources and applications are similar to that of opening the enclave by selecting the associated titles or icons on the screen.

It has already been shown how to code functionality into buttons that appear on the MCD's screen. Android allows this same coding to occur on any image displayed on the screen. This includes icons and list elements. Similarly, the coding has been discussed regarding using intents to open other activities or applications. This is exactly how the enclave functions. The user will be presented a Home activity that represents the enclave starting point after the two security methods have been performed.

The Home activity can take many different forms and will contain visual representations of the restricted applications. When the user selects one on the screen, the intent will execute per the code of the associated icon and the new application will open from inside the enclave application. These applications will contain the same signature permission as the enclave. There is no additional coding that is required beyond that which the application requires.

The user will be directed back to the enclave application every time the current restricted application is closed. Android uses the "last in, first out" stack routine to manage activities [37]. Current activities are paused when the user opens a new one. The new activity will shut down when the back button is depressed and the previous activity starts back up. This stack mechanism is also called the "back stack". The

enclave application will always be in the background, paused, when new applications are opened.

## E. SUMMARY

This section presented an architecture design for implementing a mobile security enclave. The design emphasizes solutions for providing security and access control to the enclave. There are many layers of security and control throughout this framework to include location verification, user authentication and secure application-to-application transfer. These layers prevent unauthorized access to the enclave as a whole. The next chapter concludes this work as well as provides recommendations for future efforts on mobile security enclaves.

# V. CONCLUSION AND FUTURE WORK

## A. CONCLUSIONS

This thesis proposes an architecture to allow personnel, such as first responders and military members, to securely access and manage valuable resources and applications under certain conditions. At the same time it prevents others, who are unauthorized, access to the same resources and applications. The proposed architecture lays out the blueprints for three major components of a security enclave: the location-verification application, user authentication mechanism and the security enclave construct that contains the valuable resources and applications.

The first component is the location-verification application. This is an Android application that can either check GPS coordinates or the Cell ID of the BTS to which the MCD is connected, or both. The application crosschecks this with a list containing authorized coordinates or Cell IDs before allowing the user to attempt the second form of authentication. The code for this application is explained in detail and included in this thesis.

The second component is the authentication mechanism. This component can only be accessed after being properly validated by the location verification application. This mechanism uses a challenge and response method similar to that of the GSM authentication process. The challenge is created using an algorithm (COMP128) that is compatible to the A3 algorithm used in the GSM network. This challenge is sent to the SIM card of the device where a response will be generated in similar fashion. Only authorized SIM cards will generate an appropriate response that confirms authentication.

Upon location verification and user authentication, the mobile security enclave is accessible to the user. The enclave can take many forms, but this thesis describes the framework necessary to make it application based. The enclave securely contains any

resources or applications to which the user is allowed access. The protected assets are readily accessible and transitioning from one to another is similar to that of changing activities in an Android application.

## B. FUTURE WORK

Besides the architecture described in this thesis, there are a few general areas that require further study. The first area focuses on back-end network integration, specifically pursuing other services or databases that aren't local to the device, encryption, and establishment of an update mechanism. Another area of interest is component integration to include use of removable storage devices. Finally, there are a few recommended software areas to research such as continued location verification and a method to avoid hard coding the application with authorizations. Each of these areas is described in greater detail in the following sub-sections. Addressing these concerns will ensure a product that is more robust, more flexible, and more appropriate for deployment.

### 1. Network Integration

The entire architecture is designed to operate independently of a back-end network. This is possible by installing all required resources and applications on the device. This is useful considering many occasions that require first responder support occur in environments where back-end networks are non-existent. This concept also aids in coordination amongst mobile ad-hoc networks that are not tied to dedicated infrastructures.

This does not mean that mobile secure enclaves can only exist in these types of networks. Further network integration research and programming can significantly broaden enclave use and capabilities. One such capability is to automatically push updates to all, or specific, MCDs connected to a network. Currently, the architecture requires any changes to be done manually in the application, which are then distributed manually to each device. With an automatic update method, changes to devices can occur in real-time, without user intervention, and with minimal delay in service.

Accessing databases and other services that cannot be installed on individual devices is another example of proposed network integration. Although operating autonomously has many advantages, the devices have a finite amount of processing power and memory. The ability to access resources other than what's on the device has merit but requires additional study to ensure proper access controls are in place.

Encryption of data is required when transmitting traffic between MCDs and the BTS since it is being sent through the air, susceptible to eavesdroppers. GSM currently uses the A5 algorithm for its encryption of individual packets. Further study will require incorporating the same method or one that is sufficient for the architecture, as well as the sensitivity of the information being transmitted. Commercial products are available to provide other encryption mechanisms; these may provide further confidentiality support.

### 2. Component Integration

The most important aspect of component integration is that of removable storage. It is logically possible to utilize devices such as microSD cards to store data such as authorization lists or resources for the enclave. Further research and programming is required to ensure the security is strong enough for the use of such devices. In particular, unauthorized personnel should not be able to access data on these removable storage devices.

As the proof-of-concept implementation accesses local information regarding authorized geo-locations or Cell IDs, care must be taken to ensure any production system does not allow the user to access such information directly and thereby bypass the intent of the access controls.

### 3. Software Integration

There are two functions in this architecture that need further implementation. The first is implementing continuous verification of the device location. The moment the location is verified, whether by Cell ID or GPS coordinates, the application must continuously run verification in the background to ensure the device doesn't leave the

authorized area. If it does, there should also be an, automatic, non-circumventable method, to exit the enclave.

The second function requires determining alternate means of hard coding the components of the authentication mechanisms and enclave. Currently the Cell ID list and authorized GPS coordinates are written directly into the application, as indicated above. This should be avoided, not just for security concerns, but also to make updating the enclaves a much simpler process. This alternate method should work in both cases in which there is and is not connection to a back-end network.

# APPENDIX A. MAIN ACTIVITY

```
/*****************************************
 *     Main.java
 *
 * Author:    Kevin LaFrenier
 * Date:      June 2011
 * Purpose:   Entry point for application.
 *            Calls CIDActivity & GPSActivity
 *
*****************************************/


package com.nps.lafrenier.thesis;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class Main extends Activity implements OnClickListener {
    /** Called when the activity is first created. */

       Button gpsButton, cidButton, userButton;

       @Override
       public void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              setContentView(R.layout.main);

              setConnections();
       }

       private void setConnections( ) {

              gpsButton = (Button) this.findViewById(R.id.gpsBtn);
              cidButton = (Button) this.findViewById(R.id.cidBtn);
              userButton = (Button) this.findViewById(R.id.userBtn);

              gpsButton.setOnClickListener(this);
              cidButton.setOnClickListener(this);
              userButton.setOnClickListener(this);

       }

       public void onClick(View v) {

              Button clickedBtn = (Button) v;
                     if (clickedBtn == gpsButton) {
                            // calls GPSActivity
                            Main.this.startActivity(new Intent(Main.this,
                            GPSActivity.class));
                     }
                     else if (clickedBtn == cidButton) {

                            // calls CIDActivity
                            Main.this.startActivity(new Intent(Main.this,
```

```
                        CIDActivity.class));

                } else if (clickedBtn == userButton) {

                        Main.this.startActivity(new Intent(Main.this,
                        UserIDActivity.class));
                }
        }

}
```

# APPENDIX B. MAIN XML

```
<!--****************************************
 *     main.xml
 *
 * Author:    Kevin LaFrenier
 * Date:      June 2011
 * Purpose:   Provide xml format for
 *            Main activity.  Contains three buttons
 *            and two text views.
 ****************************************-->

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
       android:orientation="vertical"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent"
       android:gravity="center">

       <TextView
              android:layout_height="wrap_content"
              android:textAppearance="?android:attr/textAppearanceLarge"
              android:id="@+id/textView1"
              android:layout_width="wrap_content"
              android:text="Location-verification"/>

       <Button
              android:id="@+id/gpsBtn"
              android:text="Get coords"
              android:textSize="10pt"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_margin="20dp"/>

       <Button
              android:id="@+id/cidBtn"
              android:text="Get CellID"
              android:textSize="10pt"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_margin="20dp"/>

       <TextView
              android:layout_height="wrap_content"
              android:textAppearance="?android:attr/textAppearanceLarge"
              android:id="@+id/textView1"
              android:layout_width="wrap_content"
              android:text="User/device Identification"/>

       <Button
              android:id="@+id/userBtn"
              android:text="Get ID"
              android:textSize="10pt"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_margin="20dp"/>

</LinearLayout>
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX C. GPS ACTIVITY

```
/****************************************
 *     GPSActivity.java
 *
 * Author:    Kevin LaFrenier
 * Date:      June 2011
 * Purpose:   GPS coordinates activity.
 *            Obtains last known GPS coordinates
 *            of device.  Displays the coords to
 *            user.  Calculates whether coords
 *            are in or out of defined location.
 *            Will call AuthActivity if they are,
 *            will close activity if they are not.
 *
****************************************/
package com.nps.lafrenier.thesis;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;

public class GPSActivity extends Activity implements OnClickListener {

	static final String BLANK = "";

	LocationManager locMgr;
	Button getLoc, authBtn, authBtn2;
	TextView LatView, LonView;
	double lat, lon, NWlat, NWlon, NElat, NElon, SElat, SElon, SWlat, SWlon;

	int precision;  //Used to truncate double

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.gps);

		setConnections();

		locMgr = (LocationManager)getSystemService(LOCATION_SERVICE);

		// Registering for location updates
		locMgr.requestLocationUpdates(LocationManager.GPS_PROVIDER, 10000,
			10000.0f, onLocationChange);
	}

	@Override
	public void onClick(View view) {

		Button clickedBtn = (Button) view;
```

```
		if (clickedBtn == getLoc) {

			// Calls the following function to output gps coords as
			// Java double into the text view "Coords"
			insertLocation();

		} else if (clickedBtn == authBtn) {

			authenticate();
		}

	}

	private void setConnections( ) {

		getLoc = (Button) this.findViewById(R.id.getLoc);
		authBtn = (Button) this.findViewById(R.id.authBtn);

		getLoc.setOnClickListener(this);
		authBtn.setOnClickListener(this);

		// text view in gps.xml, coords will be printed here
		LatView = (TextView) this.findViewById(R.id.Latitude);
		LonView = (TextView) this.findViewById(R.id.Longitude);

	}

	// Retrieves, formats & prints location
	private void insertLocation() {
		Location loc = locMgr.getLastKnownLocation
			(LocationManager.GPS_PROVIDER);

		if (loc == null) {
			Toast.makeText(this, "No location available",
				Toast.LENGTH_SHORT).show();
		} else {
			lat = loc.getLatitude();
			lon = loc.getLongitude();

			// following two lines being used to truncate double
			precision = 1000000;
			lat = Math.floor(lat * precision +.5)/precision;
			lon = Math.floor(lon * precision +.5)/precision;


			TextView myView1 = (TextView) findViewById(R.id.Latitude);
			myView1.setText("" + lat);

			TextView myView2 = (TextView) findViewById(R.id.Longitude);
			myView2.setText("" + lon);
		}

	}

private void authenticate() {

		Location loc = locMgr.getLastKnownLocation
			(LocationManager.GPS_PROVIDER);

		if (loc == null) {
			Toast.makeText(this, "No location available",
				Toast.LENGTH_SHORT).show();
		} else {
```

```
                        lat = loc.getLatitude();
                        lon = loc.getLongitude();

                        // following two lines being used to truncate double
                        precision = 1000000;
                        lat = Math.floor(lat * precision +.5)/precision;
                        lon = Math.floor(lon * precision +.5)/precision;

                        // Authorized GPS location by four corners
                        NWlat = 36.621111;
                        NWlon = -121.904167;
                        NElat = 36.621389;
                        NElon = -121.818056;
                        SElat = 36.567222;
                        SElon = -121.817500;
                        SWlat = 36.570833;
                        SWlon = -121.903056;

                        // calculation to determine location authorization
                        if ((lat < NWlat) && (lat < NElat) &&
                                    (lat > SWlat) && (lat > SElat) &&
                                    (lon > NWlon) && (lon > SWlon) &&
                                    (lon < SElon) && (lon < NElon) ) {

                              Toast.makeText(this, "Authorized",
                                    Toast.LENGTH_SHORT).show();
                              GPSActivity.this.startActivity(new
                                    Intent(GPSActivity.this,
                                    AuthActivity.class));

                        } else {

                              Toast.makeText(this, "Unauthorized",
                                    Toast.LENGTH_SHORT).show();
                              GPSActivity.this.finish( );

                        }

                  }
            }
            // Required code for registering for GPS location retrieval
            private LocationListener onLocationChange = new LocationListener() {
                  public void onLocationChanged(Location location) {
                        // required for interface, not used
                  }

                  public void onProviderEnabled(String provider) {
                        // required for interface, not used
                  }
                  @Override
                  public void onProviderDisabled(String arg0) {
                        // TODO Auto-generated method stub

                  }
                  @Override
                  public void onStatusChanged(String provider, int status, Bundle
                        extras) {
                        // TODO Auto-generated method stub
                  }
            };
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX D. GPS XML

```
<!--****************************************
 *     gps.xml
 *
 * Author:    Kevin LaFrenier
 * Date:      June 2011
 * Purpose:   Provide xml format for
 *            GPSActivity.  Contains two buttons
 *            and four text views.
 *
****************************************-->

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
       android:gravity="center"
       android:orientation="vertical"
       android:layout_height="fill_parent"
       android:layout_width="fill_parent">

       <Button
              android:id="@+id/getLoc
              android:text="Get coords"
              android:textSize="10pt"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_margin="20dp"/>

       <LinearLayout
              android:layout_height="wrap_content"
              android:layout_width="fill_parent"
              android:layout_marginLeft="20dp"
              android:layout_marginRight="20dp"
              android:layout_marginBottom="10dp">

              <TextView
                     android:text="Latitude:  "
                     android:textSize="10pt"
                     android:layout_width="wrap_content"
                     android:layout_height="wrap_content"/>

              <TextView
                     android:id="@+id/Latitude"
                     android:text=""
                     android:textSize="8pt"
                     android:layout_width="wrap_content"
                     android:layout_height="wrap_content"
                     android:textColor="#FFFFFF"/>
       </LinearLayout>

       <LinearLayout
              android:layout_height="wrap_content"
              android:layout_width="fill_parent"
              android:layout_marginLeft="20dp"
              android:layout_marginRight="20dp"
              android:layout_marginBottom="10dp">

              <TextView
                     android:text="Longitude:  "
```

```
                android:textSize="10pt"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"/>

        <TextView
                android:id="@+id/Longitude"
                android:text=""
                android:textSize="8pt"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textColor="#FFFFFF"/>
    </LinearLayout>

    <Button
            android:id="@+id/authBtn"
            android:text="Authenticate"
            android:textSize="10pt"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="20dp"/>

</LinearLayout>
```

# APPENDIX E. MANIFEST

```
<!--  ****************************************
 *     Thesis Manifest.xml
 *
 * Author:    Kevin LaFrenier
 * Date:      June 2011
 * Purpose:   Provides manifest for location-
 *            verification application.
 *
 * Permissions:
 *            ACCESS_FINE_LOCATION
 *            ACCESS_COURSE_LOCATION
 *            READ_PHONE_STATE
 *
 * Activities:
 *            GPSActivity
 *            CIDActivity
 *            AuthActivity
 *
**************************************** -->

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android=http://schemas.android.com/apk/res/android
       package="com.nps.lafrenier.thesis"
       android:versionCode="1"
       android:versionName="1.0">

       <uses-sdk android:minSdkVersion="4" />
       <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
       <uses-permission android:name="android.permission.ACCESS_COURSE_LOCATION"
              />
       <uses-permission android:name="android.permission.READ_PHONE_STATE"/>

       <application android:icon="@drawable/icon"
              android:label="@string/app_name">
              <activity    android:name=".Main"
                           android:label="@string/app_name">
                     <intent-filter>
                           <action android:name="android.intent.action.MAIN" />
                           <category android:name="android.intent.
                                  category.LAUNCHER" />
                     </intent-filter>
              </activity>
              <activity android:name="GPSActivity"/>
              <activity android:name="CIDActivity"/>
              <activity android:name="UserIDActivity"/>
              <activity android:name="AuthActivity"/>
       </application>
<manifest>
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX F. CID ACTIVITY

```
/**************************************
 *     CIDActivity.java
 *
 * Author:    Kevin LaFrenier
 * Date:      June 2011
 * Purpose:   Cell ID activity.  Obtains
 *            Cell ID of the device, displays
 *            the ID to the user and verifies it.
 *            If ID is authorized, will call
 *            AuthActivity, quits activity
 *            otherwise.  Also notifies user if
 *            no Cell ID is available.
 * ************************************/

package com.nps.lafrenier.thesis;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.telephony.gsm.GsmCellLocation;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class CIDActivity extends Activity implements OnClickListener {

	GsmCellLocation location;
	int cellID, lac;
	Button getID, authBtn, authBtn2;

	@Override
	public void onCreate(Bundle savedInstanceState) {

		super.onCreate(savedInstanceState);
		setContentView(R.layout.cid);

		setConnections();
	}

	private void setConnections() {

		getID = (Button) this.findViewById(R.id.getID);
		authBtn = (Button) this.findViewById(R.id.authBtn);

		getID.setOnClickListener(this);
		authBtn.setOnClickListener(this);


	}
	public void onClick(View v) {

		Button clickedBtn = (Button) v;
		TelephonyManager tm = (TelephonyManager)
			getSystemService(TELEPHONY_SERVICE);
```

```
                GsmCellLocation loc = (GsmCellLocation) tm.getCellLocation();
                cellID = loc.getCid();
                lac = loc.getLac();

                if (clickedBtn == getID) {

                        // cellID of a "-1" in Android means there is no CellID
                                available
                        if (cellID == -1) {
                                TextView myView1 = (TextView) findViewById(R.id.CID);
                                myView1.setText("No CellID available");
                        } else {
                                TextView myView1 = (TextView) findViewById(R.id.CID);
                                myView1.setText("" + cellID);
                        }

                        // lac of a "-1" in Android means there is no lac
                                available
                        if (lac == -1) {
                                TextView myView1 = (TextView) findViewById(R.id.LAC);
                                myView1.setText("No LAC available");
                        } else {
                                TextView myView1 = (TextView) findViewById(R.id.LAC);
                                myView1.setText("" + lac);

                        }
                } else if (clickedBtn == authBtn) {

                        // cellID of "2" is authorized, will allow user to
                                authenticate
                        if (cellID == 2) {
                                CIDActivity.this.startActivity(new
                                Intent(CIDActivity.this, AuthActivity.class));
                        // closes activity if cellID not authorized
                        } else {
                                Toast.makeText(this, "Unauthorized CellID",
                                        Toast.LENGTH_SHORT).show();
                                CIDActivity.this.finish( );
                        }
                }
        }
}
```

# APPENDIX G. CID XML

```
<!--  ****************************************
 *     cid.xml
 *
 * Author:    Kevin LaFrenier
 * Date:      June 2011
 * Purpose:   Provide xml format for
 *            CIDActivity.  Contains two buttons
 *            and four text views.
 * **************************************** -->

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
       xmlns:android="http://schemas.android.com/apk/res/android"
       android:gravity="center"
       android:orientation="vertical"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent">

       <Button
              android:id="@+id/getID"
              android:text="Get Cell ID/LAC"
              android:textSize="10pt"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_margin="20dp"/>

              <LinearLayout
                     android:layout_height="wrap_content"
                     android:layout_width="fill_parent"
                     android:layout_marginLeft="20dp"
                     android:layout_marginRight="20dp"
                     android:layout_marginBottom="10dp">
                     <TextView
                            android:text="Cell ID:  "
                            android:textSize="10pt"
                            android:layout_width="wrap_content"
                            android:layout_height="wrap_content"/>
                     <TextView
                            android:id="@+id/CID"
                            android:text=""
                            android:textSize="8pt"
                            android:layout_width="wrap_content"
                            android:layout_height="wrap_content"
                            android:textColor="#FFFFFF"/>
              </LinearLayout>

              <LinearLayout
                     android:layout_height="wrap_content"
                     android:layout_width="fill_parent"
                     android:layout_marginLeft="20dp"
                     android:layout_marginRight="20dp"
                     android:layout_marginBottom="10dp">
                     <TextView
                            android:text="LAC:  "
                            android:textSize="10pt"
                            android:layout_width="wrap_content"
                            android:layout_height="wrap_content"/>
```

```
                <TextView
                        android:id="@+id/LAC"
                        android:text=""
                        android:textSize="8pt"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:textColor="#FFFFFF"/>
            </LinearLayout>

            <Button
                    android:id="@+id/authBtn"
                    android:text="Authenticate"
                    android:textSize="10pt"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_margin="20dp"/>
</LinearLayout>
```

# APPENDIX H. AUTH ACTIVITY

```
/****************************************
 *     AuthActivity.java
 *
 * Author:    Kevin LaFrenier
 * Date:      June 2011
 * Purpose:   Authentication activity.
 *            Step-off activity for future
 *            authentication mechanism.  Contains
 *            single button with associated toast.
 ****************************************/
package com.nps.lafrenier.thesis;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class AuthActivity extends Activity implements OnClickListener {

	Button authBtn;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.auth);

		setConnections();
	}

	private void setConnections() {

		authBtn = (Button) this.findViewById(R.id.authBtn);
		authBtn.setOnClickListener(this);

	}

	public void onClick(View v) {

		Toast.makeText(this, "Success, opening enclave",
			Toast.LENGTH_SHORT).show();

	}
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX I. AUTH XML

```
<!--****************************************
 *     auth.xml
 *
 * Author:    Kevin LaFrenier
 * Date:      June 2011
 * Purpose:   Provide xml format for
 *            AuthActivity.  Contains one button
 *            and one text view.
 ****************************************-->

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
       xmlns:android="http://schemas.android.com/apk/res/android"
       android:orientation="vertical"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent"
       android:gravity="center">

       <TextView
              android:layout_height="wrap_content"
              android:textAppearance="?android:attr/textAppearanceLarge"
              android:id="@+id/textView1"
              android:layout_width="wrap_content"
              android:text="Authentication"/>

       <Button
              android:id="@+id/authBtn"
              android:text="Authenticate"
              android:textSize="10pt"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_margin="20dp"/>

</LinearLayout>
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX J. A3/A8 ALGORITHM

```
/* An implementation of the GSM A3A8 algorithm.  (Specifically, COMP128.)
 *
 * Copyright 1998, Marc Briceno, Ian Goldberg, and David Wagner.
 * All rights reserved.
 *
 * For expository purposes only.  Coded in C merely because C is a much
 * more precise, concise form of expression for these purposes.  See Judge
 * Patel if you have any problems with this...
 * Of course, it's only authentication, so it should be exportable for the
 * usual boring reasons.
 *
 *
 * This software is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.
 * Copyright remains the authors' and as such any Copyright notices in
 * the code are not to be removed.
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The license and distribution terms for any publicly available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution license
 * [including the GNU Public License.]
 */


      typedef unsigned char Byte;


      #include <stdio.h>
      /* #define TEST */


      /*
       * rand[0..15]: the challenge from the base station
       * key[0..15]: the SIM's A3/A8 long-term key Ki
```

```
 * simoutput[0..11]: what you would get back if you fed rand and key to a
 * real
 * SIM.
 *
 *   The GSM spec states that simoutput[0..3] is SRES,
 *   and simoutput[4..11] is Kc (the A5 session key).
 *   (See GSM 11.11, Section 8.16.  See also the leaked document
 *   referenced below.)
 *   Note that Kc is bits 74..127 of the COMP128 output, followed by 10
 *   zeros.
 *   In other words, A5 is keyed with only 54 bits of entropy. This
 *   represents a deliberate weakening of the key used for voice privacy
 *   by a factor of over 1000.
 *
 * Verified with a Pacific Bell Schlumberger SIM.  Your mileage may vary.
 *
 * Marc Briceno <marc@scard.org>, Ian Goldberg <iang@cs.berkeley.edu>,
 * and David Wagner <daw@cs.berkeley.edu>
 */

void A3A8(/* in */ Byte rand[16], /* in */ Byte key[16],
      /* out */ Byte simoutput[12]);

/* The compression tables. */
static const Byte table_0[512] = {
        102,177,186,162,  2,156,112, 75, 55, 25,  8, 12,251,193,246,188,
        109,213,151, 53, 42, 79,191,115,233,242,164,223,209,148,108,161,
        252, 37,244, 47, 64,211,  6,237,185,160,139,113, 76,138, 59, 70,
         67, 26, 13,157, 63,179,221, 30,214, 36,166, 69,152,124,207,116,
        247,194, 41, 84, 71,  1, 49, 14, 95, 35,169, 21, 96, 78,215,225,
        182,243, 28, 92,201,118,  4, 74,248,128, 17, 11,146,132,245, 48,
        149, 90,120, 39, 87,230,106,232,175, 19,126,190,202,141,137,176,
        250, 27,101, 40,219,227, 58, 20, 51,178, 98,216,140, 22, 32,121,
         61,103,203, 72, 29,110, 85,212,180,204,150,183, 15, 66,172,196,
         56,197,158,  0,100, 45,153,  7,144,222,163,167, 60,135,210,231,
        174,165, 38,249,224, 34,220,229,217,208,241, 68,206,189,125,255,
        239, 54,168, 89,123,122, 73,145,117,234,143, 99,129,200,192, 82,
        104,170,136,235, 93, 81,205,173,236, 94,105, 52, 46,228,198,  5,
         57,254, 97,155,142,133,199,171,187, 50, 65,181,127,107,147,226,
```

```
    184,218,131, 33, 77, 86, 31, 44, 88, 62,238, 18, 24, 43,154, 23,
     80,159,134,111,  9,114,  3, 91, 16,130, 83, 10,195,240,253,119,
    177,102,162,186,156,  2, 75,112, 25, 55, 12,  8,193,251,188,246,
    213,109, 53,151, 79, 42,115,191,242,233,223,164,148,209,161,108,
     37,252, 47,244,211, 64,237,  6,160,185,113,139,138, 76, 70, 59,
     26, 67,157, 13,179, 63, 30,221, 36,214, 69,166,124,152,116,207,
    194,247, 84, 41,  1, 71, 14, 49, 35, 95, 21,169, 78, 96,225,215,
    243,182, 92, 28,118,201, 74,  4,128,248, 11, 17,132,146, 48,245,
     90,149, 39,120,230, 87,232,106, 19,175,190,126,141,202,176,137,
     27,250, 40,101,227,219, 20, 58,178, 51,216, 98, 22,140,121, 32,
    103, 61, 72,203,110, 29,212, 85,204,180,183,150, 66, 15,196,172,
    197, 56,  0,158, 45,100,  7,153,222,144,167,163,135, 60,231,210,
    165,174,249, 38, 34,224,229,220,208,217, 68,241,189,206,255,125,
     54,239, 89,168,122,123,145, 73,234,117, 99,143,200,129, 82,192,
    170,104,235,136, 81, 93,173,205, 94,236, 52,105,228, 46,  5,198,
    254, 57,155, 97,133,142,171,199, 50,187,181, 65,107,127,226,147,
    218,184, 33,131, 86, 77, 44, 31, 62, 88, 18,238, 43, 24, 23,154,
    159, 80,111,134,114,  9, 91,  3,130, 16, 10, 83,240,195,119,253
}, table_1[256] = {
     19, 11, 80,114, 43,  1, 69, 94, 39, 18,127,117, 97,  3, 85, 43,
     27,124, 70, 83, 47, 71, 63, 10, 47, 89, 79,  4, 14, 59, 11,  5,
     35,107,103, 68, 21, 86, 36, 91, 85,126, 32, 50,109, 94,120,  6,
     53, 79, 28, 45, 99, 95, 41, 34, 88, 68, 93, 55,110,125,105, 20,
     90, 80, 76, 96, 23, 60, 89, 64,121, 56, 14, 74,101,  8, 19, 78,
     76, 66,104, 46,111, 50, 32,  3, 39,  0, 58, 25, 92, 22, 18, 51,
     57, 65,119,116, 22,109,  7, 86, 59, 93, 62,110, 78, 99, 77, 67,
     12,113, 87, 98,102,  5, 88, 33, 38, 56, 23,  8, 75, 45, 13, 75,
     95, 63, 28, 49,123,120, 20,112, 44, 30, 15, 98,106,  2,103, 29,
     82,107, 42,124, 24, 30, 41, 16,108,100,117, 40, 73, 40,  7,114,
     82,115, 36,112, 12,102,100, 84, 92, 48, 72, 97,  9, 54, 55, 74,
    113,123, 17, 26, 53, 58,  4,  9, 69,122, 21,118, 42, 60, 27, 73,
    118,125, 34, 15, 65,115, 84, 64, 62, 81, 70,  1, 24,111,121, 83,
    104, 81, 49,127, 48,105, 31, 10,  6, 91, 87, 37, 16, 54,116,126,
     31, 38, 13,  0, 72,106, 77, 61, 26, 67, 46, 29, 96, 37, 61, 52,
    101, 17, 44,108, 71, 52, 66, 57, 33, 51, 25, 90,  2,119,122, 35
}, table_2[128] = {
     52, 50, 44,  6, 21, 49, 41, 59, 39, 51, 25, 32, 51, 47, 52, 43,
     37,  4, 40, 34, 61, 12, 28,  4, 58, 23,  8, 15, 12, 22,  9, 18,
     55, 10, 33, 35, 50,  1, 43,  3, 57, 13, 62, 14,  7, 42, 44, 59,
```

```
            62, 57, 27,  6,  8, 31, 26, 54, 41, 22, 45, 20, 39,  3, 16, 56,
            48,  2, 21, 28, 36, 42, 60, 33, 34, 18,  0, 11, 24, 10, 17, 61,
            29, 14, 45, 26, 55, 46, 11, 17, 54, 46,  9, 24, 30, 60, 32,  0,
            20, 38,  2, 30, 58, 35,  1, 16, 56, 40, 23, 48, 13, 19, 19, 27,
            31, 53, 47, 38, 63, 15, 49,  5, 37, 53, 25, 36, 63, 29,  5,  7
        }, table_3[64] = {
             1,  5, 29,  6, 25,  1, 18, 23, 17, 19,  0,  9, 24, 25,  6, 31,
            28, 20, 24, 30,  4, 27,  3, 13, 15, 16, 14, 18,  4,  3,  8,  9,
            20,  0, 12, 26, 21,  8, 28,  2, 29,  2, 15,  7, 11, 22, 14, 10,
            17, 21, 12, 30, 26, 27, 16, 31, 11,  7, 13, 23, 10,  5, 22, 19
        }, table_4[32] = {
            15, 12, 10,  4,  1, 14, 11,  7,  5,  0, 14,  7,  1,  2, 13,  8,
            10,  3,  4,  9,  6,  0,  3,  2,  5,  6,  8,  9, 11, 13, 15, 12
        }, *table[5] = { table_0, table_1, table_2, table_3, table_4 };

    /*
     * This code derived from a leaked document from the GSM standards.
     * Some missing pieces were filled in by reverse-engineering a working
SIM.
     * We have verified that this is the correct COMP128 algorithm.
     *
     * The first page of the document identifies it as
     *    _Technical Information: GSM System Security Study_.
     *    10-1617-01, 10th June 1988.
     * The bottom of the title page is marked
     *    Racal Research Ltd.
     *    Worton Drive, Worton Grange Industrial Estate,
     *    Reading, Berks. RG2 0SB, England.
     *    Telephone: Reading (0734) 868601   Telex: 847152
     * The relevant bits are in Part I, Section 20 (pages 66--67).  Enjoy!
     *
     * Note: There are three typos in the spec (discovered by
     * reverse-engineering).
     * First, "z = (2 * x[n] + x[n]) mod 2^(9-j)" should clearly read
     * "z = (2 * x[m] + x[n]) mod 2^(9-j)".
     * Second, the "k" loop in the "Form bits from bytes" section is severely
     * botched: the k index should run only from 0 to 3, and clearly the
range
     * on "the (8-k)th bit of byte j" is also off (should be 0..7, not 1..8,
     * to be consistent with the subsequent section).
```

```
        * Third, SRES is taken from the first 8 nibbles of x[], not the last 8
as
        * claimed in the document.  (And the document does not specify how Kc is
        *  derived,  but  that  was  also  easily  discovered  with  reverse
engineering.)
        * All of these typos have been corrected in the following code.
        */

       void A3A8(/* in */ Byte rand[16], /* in */ Byte key[16],
              /* out */ Byte simoutput[12])
       {
              Byte x[32], bit[128];
              int i, j, k, l, m, n, y, z, next_bit;

              /* ( Load RAND into last 16 bytes of input ) */
              for (i=16; i<32; i++)
                     x[i] = rand[i-16];

              /* ( Loop eight times ) */
              for (i=1; i<9; i++) {
                     /* ( Load key into first 16 bytes of input ) */
                     for (j=0; j<16; j++)
                            x[j] = key[j];
                     /* ( Perform substitutions ) */
                     for (j=0; j<5; j++)
                            for (k=0; k<(1<<j); k++)
                                   for (l=0; l<(1<<(4-j)); l++) {
                                          m = l + k*(1<<(5-j));
                                          n = m + (1<<(4-j));
                                          y = (x[m]+2*x[n]) % (1<<(9-j));
                                          z = (2*x[m]+x[n]) % (1<<(9-j));
                                          x[m] = table[j][y];
                                          x[n] = table[j][z];
                                   }
                     /* ( Form bits from bytes ) */
                     for (j=0; j<32; j++)
                            for (k=0; k<4; k++)
                                   bit[4*j+k] = (x[j]>>(3-k)) & 1;
                     /* ( Permutation but not on the last loop ) */
                     if (i < 8)
```

```
                for (j=0; j<16; j++) {
                        x[j+16] = 0;
                        for (k=0; k<8; k++) {
                                next_bit = ((8*j + k)*17) % 128;
                                x[j+16] |= bit[next_bit] << (7-k);
                        }
                }
        }

        /*
         * ( At this stage the vector x[] consists of 32 nibbles.
         *   The first 8 of these are taken as the output SRES. )
         */

        /* The remainder of the code is not given explicitly in the
         * standard, but was derived by reverse-engineering.
         */

        for (i=0; i<4; i++)
                simoutput[i] = (x[2*i]<<4) | x[2*i+1];
        for (i=0; i<6; i++)
                simoutput[4+i] = (x[2*i+18]<<6) | (x[2*i+18+1]<<2)
                        | (x[2*i+18+2]>>2);
        simoutput[4+6] = (x[2*6+18]<<6) | (x[2*6+18+1]<<2);
        simoutput[4+7] = 0;
}


#ifdef TEST
int hextoint(char x)
{
        x = toupper(x);
        if (x >= 'A' && x <= 'F')
                return x-'A'+10;
        else if (x >= '0' && x <= '9')
                return x-'0';
        fprintf(stderr, "bad input.\n");
        exit(1);
}
```

```
int main(int argc, char **argv)
{
	Byte rand[16], key [16], simoutput[12];
	int i;

	if (argc != 3 || strlen(argv[1]) != 34 || strlen(argv[2]) != 34
			|| strncmp(argv[1], "0x", 2) != 0
			|| strncmp(argv[2], "0x", 2) != 0) {
		fprintf(stderr, "Usage: %s 0x<key> 0x<rand>\n", argv[0]);
		exit(1);
	}

	for (i=0; i<16; i++)
		key[i] = (hextoint(argv[1][2*i+2])<<4)
			| hextoint(argv[1][2*i+3]);
	for (i=0; i<16; i++)
		rand[i] = (hextoint(argv[2][2*i+2])<<4)
			 | hextoint(argv[2][2*i+3]);
	A3A8(key, rand, simoutput);
	printf("simoutput: ");
	for (i=0; i<12; i++)
		printf("%02X", simoutput[i]);
	printf("\n");
	return 0;
}
#endif
```

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF REFERENCES


[1] Wikipedia, "GSM," August 18, 2011. http://en.wikipedia.org/wiki/GSM

[2] GSM World, "History," April 9, 2011. http://www.gsmworld.com/about-us/history.htm

[3] GSM World, "History," April 9, 2011. http://www.gsmworld.com/about-us/history.htm

[4] ICT Statistics Newslog, "Global Mobile Phone Subscribers To Reach 4.5 Billion by 2012," March 11, 2008. http://www.itu.int/ITU-D/ict/newslog/Global+Mobile+Phone+Subscribers+To+Reach+45+Billion+By+2012.aspx

[5] ZTE, "Global GSM Incremental Market Analysis," April 19, 2010. http://wwwen.zte.com.cn/endata/magazine/ztetechnologies/2010/no4/articles/201004/t20100419_182951.html

[6] L. O. Walters and P. S. Kritzinger, "Cellular Networks: Past, Present, and Future," *Crossroads*, Vol. 7, Issue 2, pp. 5, 2000.

[7] K. Vedder, "GSM: Security, Services, and the SIM," *Computer Science*, 1528, pp. 227, 1998.

[8] Osmocom, "OpenBSC," April 12, 2011. http://openbsc.osmocom.org/trac/wiki/OpenBSC

[9] Osmocom. "osmo-nitb," April 12, 2011. http://openbsc.osmocom.org/trac/wiki/osmo-nitb/

[10] A. Mehrotra and L. S. Golding, "Mobility and Security Management in the GSM System and Some Proposed Future Improvements," *Proceedings of the IEEE*, Vol. 86, Issue 7, pp. 1483, 1998.

[11] Y. J. Choi and S. J. Kim, "An Improvement on Privacy and Authentication in GSM," *Computer Science*, 3325, pp. 17, 2005.

[12] Y. J. Choi and S. J. Kim, "An Improvement on Privacy and Authentication in GSM," *Computer Science*, 3325, pp. 15, 2005.

[13] L. Pesonen, "GSM Interception," *White Paper*, University of Technology, Helsinki, pp. 2, 1999.

[14] A. Schoffl and M. Irger, “Communication Infrastructure: GSM Communication,” Johannes Kepler Universitat Linz, 2001.

[15] A. Mehrotra and L. S. Golding, “Mobility and Security Management in the GSM System and Some Proposed Future Improvements,” *Proceedings of the IEEE,* Vol. 86, Issue 7, pp. 1489–1491, 1998.

[16] Wikipedia. “List of Digital Distribution Platforms for Mobile Devices,” August 18, 2011. http://en.wikipedia.org/wiki/List_of_digital_distribution_platforms_for_mobile_devices

[17] Android. “What is Android?” August 5, 2011. http://developer.android.com/guide/basics/what-is-android.html

[18] Android. “What is Android?” August 5, 2011. http://developer.android.com/guide/basics/what-is-android.html

[19] Android. “What is Android?” August 5, 2011. http://developer.android.com/guide/basics/what-is-android.html

[20] LGS. *Hardware Manual for Release 3.5 Tactical Base Station Router,* Issue 4.2. January 2009.

[21] Android, “Package android.Location,” August 5, 2011. http://developer.android.com/reference/android/location/package-summary.html

[22] Android, “Public static final class Manifest.permission,” August 5, 2011. http://developer.android.com/reference/android/Manifest.permission.html

[23] 3rd Generation Partnership Project (3GPP). *Technical Specification Group System and Service Aspects; Security Related Network Functions,* TS 43.020, V10.0.0, pp. 20, 2011-03.

[24] K. Vedder, “GSM: Security, Services, and the SIM,” *Computer Science,* 1528, pp. 226, 1998.

[25] GSM World, “GSM Security Algorithms,” June 30, 2011. http://www.gsmworld.com/our-work/programmes-and-initiatives/fraud-and-security/gsm_security_algorithms.htm

[26] 3rd Generation Partnership Project (3GPP). *Technical Specification Group System and Service Aspects; Security Related Network Functions,* TS 43.020, V10.0.0, pp. 50–51, 2011-03.

[27] Philipp Sudmeyer, “A Performance Oriented Implementation of COMP128,” Ruhr-University Bochum, pp. 8, 2006.

[28] K. Vedder, “GSM: Security, Services, and the SIM,” *Computer Science*, 1528, pp. 232–233, 1998.

[29] K. Vedder, “GSM: Security, Services, and the SIM,” *Computer Science*, 1528, pp. 233, 1998.

[30] ETSI, “Digital Cellular Telecommunications System (Phase 2+); Specification of the Subscriber Identity Module – Mobile Equipment (SIM-ME) Interface,” GSM 11.11, pp. 33, 1995.

[31] Secure Element Evaluation Kit (SEEK) for the Android Platform, “The SmartCard API,” July 1, 2011. http://code.google.com/p/seek-for-android/

[32] Secure Element Evaluation Kit (SEEK) for the Android Platform, “UICC Support,” July 2, 2011. http://code.google.com/p/seek-for-android/wiki/UICCSupport

[33] Android, “Application Fundamentals,” August 5, 2011. http://developer.android.com/guide/topics/fundamentals.html

[34] Android, “Activities,” August 5, 2011. http://developer.android.com/guide/topics/fundamentals/activities.html

[35] Android, “Intents and Intent Filters,” August 5, 2011. http://developer.android.com/guide/topics/intents/intents-filters.html

[36] Android, “Security and Permissions,” August 5, 2011. http://developer.android.com/guide/topics/security/security.html

[37] Android, “Activities,” August 5, 2011. http://developer.android.com/guide/topics/fundamentals/activities.html

THIS PAGE INTENTIONALLY LEFT BLANK

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Lieutenant Colonel Vaughn Pangelinan, USMC
   Naval Postgraduate School
   Monterey, California

4. Director, Training and Education, MCCDC, Code C46
   Quantico, Virginia

5. Director, Marine Corps Research Center, MCCDC, Code C40RC
   Quantico, Virginia

6. Marine Corps Tactical Systems Support Activity (Attn: Operations Officer)
   Camp Pendleton, California

7. Marine Corps Systems Command (MARFORSYSCOM)
   Program Group 11 MAGTF C2 Systems
   Quantico, Virginia

8. Dr. Gurminder Singh
   Naval Postgraduate School
   Monterey, California

9. Mr. John H. Gibson
   Naval Postgraduate School
   Monterey, California